NPS ARCHIVE
1997.12
PRENGER, K.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COSTS AND BENEFITS OF SOFTWARE PROCESS IMPROVEMENT

By

Karen D. Prenger

December, 1997

Thesis Advisor: James Emery
Second Reader: Elizabeth Gramoy

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302 and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE December 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE COSTS AND BENEFITS OF SOFTWARE PROCESS IMPROVEMENT	5. FUNDING NUMBERS
6. AUTHOR Prenger, Karen D.	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSOR/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense of the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

There are numerous problems in DoD software development projects. The ad hoc practices used in the military services and in industry have resulted in unpredictable costs and schedules and low-quality products. This thesis proposes that one solution to these problems is to integrate Software Process Improvement (SPI) activities based on a proven model into software development projects. Both a formal and an informal approach to SPI will be discussed. The thesis will also describe not only the problems encountered in most software development projects, but also the activities defined in these SPI approaches that are designed to solve these problems. A case study of a military project that has spent several years implementing SPI activities based on Software Engineering Institute's (SEI) Capability Maturity Model (CMM) is presented. The SPI activities were implemented in an effort to deliver a high quality product with high reliability while maintaining a high level of control of costs and schedule. This project has succeeded in its goals and the costs and benefits of the project's efforts will be presented.

14. SUBJECT TERMS Software Process Improvement, Software Engineering Institute's Capability Maturity Model, SmartNet Project, Rapid Application Development	15. NUMBER OF PAGES 110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT **UNCLASSIFIED**	18. SECURITY CLASSIFICATION OF THIS PAGE **UNCLASSIFIED**	19. SECURITY CLASSIFICATION OF ABSTRACT **UNCLASSIFIED**	20. LIMITATION OF ABSTRACT **UL**

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std 239-18

Approved for public release; distribution is unlimited

COSTS AND BENEFITS OF SOFTWARE PROCESS IMPROVEMENT

Karen D. Prenger
B.S., National University, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1997

NPS ARCHIVE
1997.12
PRENGER, K.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

ABSTRACT

There are numerous problems in DoD software development projects. The ad hoc practices used in the military services and in industry have resulted in unpredictable costs and schedules and low-quality products. This thesis proposes that one solution to these problems is to integrate Software Process Improvement (SPI) activities based on a proven model into software development projects. Both a formal and an informal approach to SPI will be discussed. The thesis will also describe not only the problems encountered in most software development projects, but also the activities defined in these SPI approaches that are designed to solve these problems. A case study of a military project that has spent several years implementing SPI activities based on Software Engineering Institute's (SEI) Capability Maturity Model (CMM) is presented. The SPI activities were implemented in an effort to deliver a high quality product with high reliability while maintaining a high level of control of costs and schedule. This project has succeeded in its goals and the costs and benefits of the project's efforts will be presented.

TABLE OF CONTENTS

I. INTRODUCTION.....1

II. ISSUES INVOLVED IN SOFTWARE DEVELOPMENT.....5

III. WHY IS SOFTWARE DEVELOPMENT SO PROBLEMATIC?.....9

IV. AN ANALYSIS OF TWO DIFFERENT APPROACHES TO SOFTWARE DEVELOPMENT.....13

V. CAN PROCESSES SOLVE THE PROBLEMS OF SOFTWARE DEVELOPMENT?.....19

VI. SOFTWARE PROCESS IMPROVEMENT EFFORTS IN INDUSTRY.....21

VII. SOFTWARE PROCESS IMPROVEMENT EFFORT AT SPACE AND NAVAL WARFARE SYSTEMS CENTER, SAN DIEGO.....25

VIII. DESCRIPTION OF THE SMARTNET PROJECT 27

IX. SOFTWARE PROCESS IMPROVEMENT ACTIVITIES IMPLEMENTED ON SMARTNET.....31

X. ANALYSIS OF COSTS OF SOFTWARE PROCESS IMPROVEMENT ACTIVITIES IMPLEMENTED.....37

XI. ANALYSIS OF BENEFITS OF SOFTWARE PROCESS IMPROVEMENT ACTIVITIES IMPLEMENTED.....41

XII. CONCLUSIONS.....49

APPENDIX A. THE SOFTWARE CHANGE REQUEST.....51

APPENDIX B. ACTUAL VERSUS ESTIMATED LEVEL OF EFFORT REPORT.....53

APPENDIX C. CLOSED ACTION ITEMS/SOFTWARE CHANGE REQUESTs REPORT.....55

APPENDIX D. SOFTWARE CHANGE REQUESTS/ACTION ITEMS CLOSED BETWEEN <START/END DATE>.....57

APPENDIX E. TEAM POWER REPORT.....63

APPENDIX F. SOFTWARE CHANGE REQUEST METRICS FOR PROBLEM SOFTWARE CHANGE REQUESTS.....65
APPENDIX G. VERSION DESCRIPTION DOCUMENT.....85
LIST OF REFERENCES.....91
BIBLIOGRAPHY.....93
INITIAL DISTRIBUTION LIST.....95

LIST OF FIGURES

Figure 1. SmartNet Software Change Request Level of Effort.....45
Figure 2. Estimated Costs of Fixing Software Change Requests.....46
Figure 3. SmartNet Software Change Requests.....47

LIST OF TABLES

Table 1. Processes Designed to Solve Software Development Problems.....20
Table 2. Summary of Costs Incurred Implementing SPI Activities.....39
Table 3. SmartNet Software Release Dates.....42
Table 4. Defect Rates of SmartNet Software Versions.....48

ACKNOWLEDGMENT

I would especially like to thank my thesis advisors, Professor James Emery for his guidance, and Elizabeth Gramoy for her help and encouragement. I would like to thank the dedicated people that work in the Software Engineering Process Office (SEPO) and as Software Process Improvement (SPI) agents at Space and Naval Warfare Systems Center, San Diego (SSC-SD) for helping with reference material and providing ideas. A special thanks also goes to the SmartNet team who went out of their way to provide reports and information for the case study that is a major part of this thesis.

I. INTRODUCTION

Today's major defense systems depend largely on the quality of complex and increasingly costly software. Many major weapon systems cannot operate if the software fails to function as required. Because software errors can cause a system to fail, possibly with life-threatening consequences, software-intensive systems need to be thoroughly tested before production. Since the early 1970s, the General Accounting Office (GAO) has reported problems in operational test and evaluation of defense acquisition programs. Senior DoD officials, as well as Members of the Congress, are concerned that many of these problems continue today, particularly in software-intensive systems. [Ref. 1]

DoD software costs totaled over $30 billion a year in 1993 (estimated to be $42 billion by 1995), of which about two-thirds was for maintaining, upgrading, and modifying operational systems already in production. In contrast, the cost of computer hardware components that are integral to weapon systems and other critical military and intelligence systems are expected to remain stable at about $6 billion annually between 1990 and 1995. [Ref. 1]

According to a 1992 report by the Secretary of the Air Force, virtually all software-intensive defense systems suffer from difficulties in achieving cost, schedule, and performance objectives. It has been repeatedly demonstrated during operation testing and, in some cases, during operations in the field, that these systems do not meet user requirements. Most of these software problems could have been identified and addressed during earlier development testing. [Ref. 1]

In December 1992, GAO reported that DoD's mission-critical computer systems continued to have significant software problems due in part to a lack of management attention, ill-defined requirements, and inadequate testing. [Ref. 1]

A 1979 GAO report states: [Ref. 2]

- > 50% of contracts had cost overruns
- > 60% of contracts had schedule overruns
- > 45% of software could not be used
- > 29% of software was never delivered
- > 19% of software had to be reworked

- ~ 3% of software had to be modified
- < 2% of software was usable as delivered

An unpublished review of 17 major DoD software contracts found that the average 28-month schedule was missed by 20 months. One four-year project was not delivered for seven years; no project was on time. Deployment of the B1 bomber was delayed by a software problem, and the $58 billion A12 aircraft program was canceled partly for the same reason. [Ref. 3]

One recent GAO report summarized more than 20 GAO case studies involving software or software-related problems in the military and concluded, "The understanding of software as a product and of software development as a process is not keeping pace with the growing complexity and software dependence of existing and emerging mission-critical systems." [Ref. 3]

The problems in DoD software development projects are numerous. The issues involved and some of the most common problems encountered in software development will be discussed in the next few sections of the thesis. I propose that one solution to these problems is to integrate Software Process Improvement (SPI) activities based on a proven model into software development projects. The model that is used at Space and Naval Warfare Systems Center San Diego (SSC-SD) is the Software Engineering Institute's (SEI) Capability Maturity Model (CMM). A less formal approach, called Rapid Application Development (RAD) will also be discussed as a solution to improving small software projects. This thesis also contains a case study of a project at SSC-SD using the CMM.

Based on a consensus of many experts and much literature in this field, an organization's chance for success depends first on having an exceptional manager and an effective development team (PEOPLE). Secondly, it depends on its effective use of TECHNOLOGY, and finally, on its PROCESS maturity. [Ref. 4] In a software organization:

PEOPLE refers to the attributes of the personnel responsible for managing, performing, or overseeing the development and maintenance of their software products.

Management commitment and ability to hire and retain competent people are the most crucial elements in predicting an organization's success.

TECHNOLOGY refers to the tools, languages, information, applications, and environments needed to develop and maintain software.

PROCESS refers to the way people approach software development and maintenance. Process is a particular method of doing something, generally involving a number of steps or operations. A software process consists of methods, activities, plans, practices, procedures, and steps used to produce and maintain software. It is the fiber that connects people to technology and allows them to effectively use their technology. Process maturity is how well a process is defined, managed, measured, and controlled, and how effective it is. Software process maturity is an indicator of software development capability. The quality of a software system is governed by the quality of the process used to develop it.

Of the three, talented people are, by far, the most important element of any software organization. But organizations must strive for balance between good software engineering processes; the proper, although not necessarily the most current technology; and competent management of their workforce. [Ref. 4]

II. ISSUES INVOLVED IN SOFTWARE DEVELOPMENT

According to [Ref. 5], a system is defined as "*an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective.*" The elements making up the system are a database, documentation, people, procedures, software, firmware, and hardware. Some of the software development issues are:

Selecting an organization and team structure. As stated in [Ref. 2], the purpose of an organizational structure is to focus the efforts of many to a selected goal. The program/project manager selects the optimum project structure for the project environment. That structure could be *project, matrix*, or *functional*. The *project structure* keeps people involved through the life of the project. It provides more continuity and gives the project manager maximum control. The *matrix* organization takes people from other projects/divisions within the organization to form a new team that exists as long as needed to get the project done. The advantages of the *matrix* organization are that it provides flexibility in staffing and the project manager does not have to manage the same people too long. The disadvantages are that there could be problems allocating resources and coordinating efforts among changing team members. Another disadvantage is the employee could end up with two supervisors. In the *functional structure,* each team is assigned one or more functional tasks (such as writing the user interface) and each team has a specific structure. An advantage is that it is easier to stay up to speed technically in a functional structure; the disadvantage is that it is difficult to coordinate among multiple functions. Also, employees resist staying on the project too long.

As described in [Ref. 2], there are three types of team structures:

- Democratic decentralized (DD): This team has no permanent leader. Decisions are made by group consensus and communications are horizontal. People that work on this type of team tend to have a high morale and high job satisfaction. This team is communication intensive.

- Controlled decentralized (CD): This team has a defined leader who coordinates specific tasks and has secondary leaders who have responsibility for subtasks. This team is more structured than the DD.

- Controlled centralized (CC): a team leader manages top-level problem solving and internal team coordination. This team is very structured.

The best team structure depends on project factors such as difficulty of the problem, size of the program, team lifetime, degree to which the problem can be modularized, required quality and reliability, rigidity of the delivery date, and degree of communication required for the project. The DD structure is good for small (five people or fewer) projects due to the time needed for a high level of communication. This structure is also good for very difficult projects that may need more time for creative ideas and problem solving to occur. The CD structure is good for larger projects that are more modular, require high reliability, and have a shorter schedule to meet. The CC structure is also good for larger, less difficult, highly modular projects. The CC structure is desirable when there are strict time constraints.

Staffing the project. Staffing the project is extremely important. The people selected to work on a project can make the project a success or a failure. It takes time to evaluate applications and interview personnel. Certainly, it would be desirable to put all the top software engineers together for the best possible team. However, that is not always feasible. Each organization has junior people who need training and mentoring. To put them all together on a software development project without senior technical staff to help could be a disaster for the project. Ideally, the team would be a mixture of new, inexperienced people and *seasoned* software developers. The developers with experience can help and mentor the less experienced people on the team. Training should be provided as needed.

Sometimes it might be difficult to attract the best people for the job. A project manager is in competition with other projects within the organization for the best people. Assuming the pay, benefits, and location are the same throughout the organization, one way to attract the best people is to better yourself as a manager. [Ref. 2]

Another important aspect to consider after the team has been selected and staffed is to make sure the manager/technical lead solves the personnel problems as they occur without letting them hurt the project/team morale. For example, if the manager keeps a poor performer on the team, eventually the other members will lose incentive to work and it hurts the manager's reputation.

Selecting a software model/methodology. A methodology refers to the standards and procedures that affect the planning, analysis, design, development, implementation, operation, support, and disposal of a software-intensive system. The software life cycle management methodologies include evolutionary, incremental, waterfall, spiral, or any other tailored method applicable to the environment. A methodology should be chosen based on the nature of the program, software domain, the methods and tools used, and the controls and deliverables required. [Ref. 5]

Choosing an appropriate life cycle methodology is not always an easy task. Each methodology has advantages and disadvantages. Current guidance for MIS development is that the incremental or evolutionary methods constitute more effective risk management and provide earlier satisfaction of user requirements.

The **evolutionary life cycle** is an alternative strategy for systems where future requirement refinements are anticipated (i.e., the requirements evolve over time) or where there is medium to high technical risk. This method involves building prototypes for the user to test and refine the requirements. This is similar to the spiral model in that the development process is iterative. This method is well-suited for high technology software-intensive systems where requirements beyond the core capability can generally be identified.

The **incremental life cycle management method** involves developing a software-intensive product in a series of increments of increasing functional capability. The requirements of the system are identified at the beginning of the project. Benefits of this methodology are:

- Risk is spread across several smaller increments instead of one large development.
- Requirements are stabilized during the production of a given increment.

- Understanding the requirements for later increments becomes clearer based on the user's ability to gain a working knowledge of earlier increments.

The incremental method is most appropriate for low-to medium-risk programs.

The **waterfall model** (also referred to as "Grand Design") was the first to formalize a framework for software development phases, and placed emphasis on up-front requirements and design activities and on producing documentation during early phases. The major drawback is its inherent sequential nature. Any attempt to go back two or more phases to correct a problem would result in major increases in cost and schedule. It is not suited for modern development techniques such as prototyping and automatic code generation. It also is not suited for unprecedented systems because it inhibits flexibility.

The **spiral method** provides a risk-reducing approach to the software life cycle. It combines basic waterfall building block and evolutionary/incremental prototype approaches to software development. The building block activities include the preliminary, detailed, and critical design reviews, code, unit test, integration and test, and qualification test. The advantages of the spiral model are its emphasis on procedures, such as risk analysis, and its adaptability to different life cycle approaches.

Selecting/defining the processes that will be used by the team. The project manager should use processes to perform the major functions of software development (planning, requirements analysis, software design, coding, unit testing, unit integration and testing, configuration management, product evaluation, quality assurance). One of the more formal approaches to processes is to implement policies and processes based on an accepted model. Examples would be the CMM, and the ISO 9000 model. The models can be tailored to be as formal or informal as the project manager desires.

Selecting tools to be used by the team. This is an important part of the planning phase of the project. The tools refers to the hardware and software used by the team to design, develop, debug, and document the software project, as well as perform configuration management and maintenance functions. The tools selected need to be functional, easily acquired, affordable, and easy to learn how to use.

III. WHY IS SOFTWARE DEVELOPMENT SO PROBLEMATIC?

In light of all the issues just described with software development, it comes as no surprise that there are so many problems encountered when developing software. The Department of the Air Force states in [Ref. 5] that overwhelming evidence indicates that software programs fail for the following common reasons:

Software's inherent complexity. Software is risky because it is hard to build. The complexity of hardware pales in comparison with the complexity of software. Complexity plagues us because we often fail to take a disciplined approach to design and create more complexity than needed. Software complexity should be kept to a minimum. Highly complex solutions are destined to be high cost maintenance nightmares! The more complex the software, the more difficult it is to understand and the greater the chance for defects to propagate throughout the code. The cost of making changes and correcting defects often soars beyond acceptable levels, resulting in programs being abandoned after exorbitant expenditures of unrecoverable resources.

Our inability to estimate cost, schedule, and size. The fundamental reason software-intensive development projects overrun cost and schedule, with resulting quality and performance shortfalls, is our inability to estimate. No matter how smooth our development process, how efficient our tools, or how smart our designers, our predictions of cost and schedule are frequently out of sync with what actually occurs in the production of a software product. We often forget that software development involves much more than simply writing code. For example, we are still learning that software maintenance consumes from 60% to 80% of our software dollars. We also do not account for the amount of scrap or rework of code involved when a developer has an ad hoc, chaotic development process, the cost of which Boehm claims to be about 44% of every dollar spent. Predicting the size and complexity of the software to be built is at the heart of our estimation deficiencies. When a software development is precedented (i.e., a similar system has been developed), size and complexity projections (and thus cost and schedule) are usually more accurate. In unprecedented systems, however, our ability to estimate the intangible is low.

Programs tend to get in trouble in small, progressively compounding increments. When the product is late (and/or over cost), we apply management pressure to reduce the slack between our projected delivery date and the illusive real one. This aggravates the problem into a *catch-22* situation. With inadequate resource and schedule estimates, the time required to *build-quality-in* may be insufficient. To meet schedule and keep down cost, the next easiest thing to cut is testing. Before we realize it, a late, over-cost program evolves into an unreliable one. When a cost/schedule disaster is discovered, developers often try to protect their contract through alternative proposals that attempt to deliver less for the same price. This leads to down-scoping, or eliminating requirements, in an attempt to stay within initial projections. This is a very serious situation because it means resources have been expended, often exhausted, and the user does not get the system for which they paid. There are many cases of programs that have been canceled without the delivery of a single operational product after years of schedule and cost overruns.

There is also a problem with optimistic estimates. In DoD projects are subject to spending and budgeting scrutiny from the Congress, the press, and upper management. Under pressure, contractors and military managers often make overly optimistic estimates about how much the software will cost and how long it will take to produce. The pessimistic cost, schedule, and size estimates are often discarded and our projections are based on the *best of all possible worlds*. Risk is not managed and a management reserve or a worst-case scenario is not built into the cost and schedules for fear the program will not get funded or approved if more realistic figures are submitted. This increases the likelihood for shortcuts in the development of a product that was improperly funded and scheduled.

Unstable Requirements. One big cause of software program failures, upon which all the reports and studies undeniably concur, is requirements instability. It is reasonable to expect that requirements of the software system are going to change because user needs change, and when building weapon systems, the world and threats can change. One way to prepare for the changes is to build software systems with an architecture that tolerates changing requirements without compromising design. Another way to handle the changes is to control how, and at what pace, inevitable requirements changes are

incorporated. If ad hoc, sporadic, or frequent modifications to requirements or their interpretation are inflicted on developers, changes in cost and schedule are a given. Sometimes what appear to be minor changes have dramatic side effects elsewhere in the software. If full (technical and effort) evaluation of change consequences are not included in the management process, the incremental incorporation of changed requirements can invalidate estimates of cost and schedule, diminishing product quality.

One potential source of instability is inadequately stated requirements. Indefinite and undefined software requirements also lead to cost and schedule changes, which can continue even after the program enters development. Requirements definition and analysis is the most important task and is also very difficult for all sectors of industry and government.

Misinterpretation of customer/user requirements is a major, if not the greatest, contributor to software failure. Not understanding customer's needs and/or inadequately stating requirements have often been the source of costly support problems and ultimate program failures. It is critical to get customer/user feedback throughout the project to determine whether perceived user needs have been correctly translated into software functionality.

The main reason errors occur during requirements definition and analysis is lack of communication. During the requirements phase, the user tries to articulate a concept of the system functionality and performance required. The software engineer attempts to translate user definitions into models of information, control flow, operational behavior, and data content. The chances for misinterpretation are high.

Data collected at Rome Laboratory indicate that over 50% of all software errors are "requirements errors." Requirements errors are more expensive to correct the further they percolate throughout the life cycle.

Poor Problem Solving/Decision Making by Management. Management is, like all other activities in software development, a problem-solving exercise. It involves deciding what must be accomplished, how to do it, monitoring what is being performed, and evaluating what has occurred. The "what" is expressed in the Software Development

Plan. The "how" is an allocation of resources (people, lab facilities, computers, tools) to get the job done within schedule and budget.

It's easy to forget that software development is dynamic and the original plans and estimates must be updated as requirements change or people leave the project. When change requests are submitted, it is a common mistake to fail to make a solid estimate of their impact on the cost and schedule estimates. It is also a common mistake to fail to demand additional payment for additional functionality, in an effort to please the customer.

IV. AN ANALYSIS OF TWO DIFFERENT APPROACHES TO SOFTWARE DEVELOPMENT

A Formal Approach. The DoD established the SEI in 1984 to advance the practice of software engineering. The mission of the SEI is to provide leadership in advancing the state of the practice of software engineering to improve the quality of systems that depend on software. The mission of the SEI's Software Process Program is to provide leadership in assisting software organizations to develop and continuously improve their capability to identify, adopt, and use sound management and technical practices. [Ref. 3]

The CMM for Software, developed by the SEI, is a framework that describes the key elements of an effective software process and provides guidance on how to establish and improve software development processes. The CMM describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one. Five levels of maturity encompass this path. [Ref. 3]

The CMM presents recommended practices in a number of Key Process Areas (KPAs) that have been shown to enhance software development and maintenance capability. When followed, these practices improve the ability of organizations to meet goals for cost, schedule, functionality, and product quality. [Ref. 3]

The CMM contains the following KPAs in maturity levels two and three: [Ref. 6]

- *Requirements management* is used to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project.
- *software project planning* is used to establish reasonable plans for performing the software engineering and for managing the software project.
- *software project tracking and oversight* is used to establish adequate visibility into actual progress so that management can take effective actions when the project deviates significantly from the plans.
- *software subcontract management* is used to select qualified software subcontractors and manage them effectively.

- *software quality assurance* is used to provide management with appropriate visibility into the process being used by the software project and of the products being built.
- *software configuration management* is used to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.
- *organization process focus* is used to establish the organizational responsibility for software process activities that improve the organization's overall software process capability.
- *organization process definition* is used to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.
- *training program* is used to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently.
- *integrated software management* is used to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets, which are described in Organization Process Definition. This tailoring is based on the business environment and technical needs of the project.
- *software product engineering* is used to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
- *intergroup coordination* is used to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.
- *peer reviews* is used to remove defects from the software work products early and efficiently.

Correlation between project success and using the CMM was established in a study done by the Air Force Institute of Technology in September 1995. An excerpt from

their report says, "The aim of our research was to determine the nature of a correlation between the CMM rating and software development success. We were able to show correlation between CMM rating and the cost and schedule performance.... We observed improved cost and schedule performance with increasing process maturity. Specifically, the least mature organizations were likely to have difficulty adhering to cost and schedule baselines...the more mature organizations were likely to have on-baseline cost and schedule performance. This study has validated a correlation between project success and CMM ratings...". [Ref. 7]

According to SEI, all of the military services have used ad hoc practices that have resulted in unpredictable costs and schedules and low-quality software products that do not meet users' needs. To address these problems, the services have taken different approaches to improving software development and test and evaluation and are in various stages of implementing those improvements. [Ref. 1]

An Informal Approach. The most publicized software development failures are usually large projects either in military or industry. Less conspicuous are the far more numerous smaller projects that take too long, cost too much, and deliver crummy stuff. [Ref. 8] Dr. James Emery has been closely involved in a number of successful development projects. They all followed a similar approach to a process called *Rapid Application Development (RAD)*: a small team, working about a year in a relatively informal environment, delivered a successful system that matched or exceeded management expectations. Although large projects need highly disciplined processes, this approach seems to work well for small to medium projects. [Ref 8, 9]

The RAD process has three essential ingredients:

- an *adaptive methodology* that can take advantage of the organizational learning that takes place during the development process
- a *productive set of development tools* that makes it feasible to respond quickly to user feedback
- a *small team of highly competent developers*

These ingredients were also described in the Introduction of this thesis and in [Ref. 4]. The basic idea of an *adaptive methodology* is to proceed through an iterative

process of design that eventually converges to an acceptable design. Each iteration is a working prototype. This prototype enables maximum communication between users and developers. It is much easier to comprehend how the system will work if you have a prototype to test as opposed to a design document to read. Eventually the evolving prototype becomes functional and robust enough to deploy as a production system. Once in production, revisions occur at a much slower pace. Initial planning of the tasks involved and the data model to be used is an important part of the adaptive process and will contribute greatly to the success of the project.

A *set of productive development tools* is the second ingredient to this approach to software development. The tools must allow for a seamless migration through a series of prototypes to the eventual production system. Depending on the project environment, the tool set would consist of several complementary tools. Tools needed could include ones used to do project scheduling and planning, writing reports, generating forms, maintaining a database, and project accounting.

A *small team of highly talented developers* is the third ingredient to this approach. If the project is small enough, most of the management and coordination overhead required on a large project can be avoided. Communication is not limited by hierarchical reporting relationships. Each member is encouraged to contribute his or her ideas in open dialogues. This type of team was described earlier in section two of this thesis. The high level of communication and opportunity for creativity are two of the greatest strengths of a small team.

The Tradeoffs. Some of the drawbacks to implementing the formal approach to software development and software process improvement are described in the SmartNet case study found later in the thesis. It takes a lot of time and money to develop/tailor the organizational processes to be used by the projects and to train the employees on how to use these processes. Depending on the culture of the organization, getting management and the employees to "buy-in" to this process may or may not be easy. It also takes time (sometimes two to three years) to see the benefits of SPI activities. However, when faced with managing and organizing a large software development project, the formal processes provide management control over the development process and help guarantee a reliable,

high quality system, delivered on time. One way for new projects to realize cost savings is to leverage off of the lessons learned from projects that have more experience with SPI efforts.

The informal RAD approach is not appropriate for large projects or projects where program reliability and maintainability (or lack of it) can endanger lives (such as in military weapons systems). A formal approach is needed in these instances. However, the RAD approach has certainly been successful on numerous small projects.

The initial costs of implementing a RAD approach would be in software/hardware tools versus costs of implementing processes and training employees in the formal approach.

An adaptive development process conflicts with the school of software engineers who prize well-defined up-front requirements and a disciplined, reproducible implementation process. They would no doubt find the adaptive development process undisciplined and sloppy, seriously lacking according to CMM criteria. With an adaptive process, one does not know up front exactly what the end result will be; instead, one arrives at the end result through a process of organizational learning and discovery. It is an uncertain and ambiguous venture. [Ref. 8]

The adaptive development process approach is one way to plan for the changes in requirements that will inevitably occur during the development of a software system. The CMM attempts to plan for requirements changes by implementing a requirements management plan, a configuration management plan, and a project tracking and oversight plan.

V. CAN PROCESSES SOLVE THE PROBLEMS OF SOFTWARE DEVELOPMENT?

In this section I will look at how both the formal and informal processes described earlier can help solve the problems encountered on software development projects.

As mentioned previously in Section III of this thesis, the main problems occurring in software development projects are due to:

- program complexity
- ability to estimate cost, schedule, and size
- unstable requirements
- poor problem solving/decision making by management

Problems can also occur during the planning and staffing phases of the software project as described in Section II of this thesis.

Table 1 lists the problems that can occur during the development of a software project and the process that was designed to help solve the problem for both the formal and informal approach to software development.

The CMM key process areas that are listed in Table 1 were described earlier in Section II of the thesis. The informal approach to software development discussed earlier, RAD, attempts to solve the problems of software complexity and unstable requirements by implementing an adaptive methodology. This methodology uses prototypes to develop the requirements and keep a high level of communication between the developer and the users. Since it is expected that the requirements will change, this process plans for these changes.

The RAD approach uses productive development tools to help solve the problems of estimating cost and schedule of the program. Time boxing refers to the practice of setting a time limit on an activity and using what you have developed so far when the time limit arrives. The tools can also aid in communication, which in the long run, enables better problem solving and decision making. Since this approach would be used on small to medium projects, a lot of the management problems would be eliminated due to the size of the project team.

Regardless of the methodology followed for software development, there will always be problems because industry and government are never satisfied with their technical achievements and are always *pushing the technical envelope.*

PROBLEMS ENCOUNTERED	**CMM KPA DESIGNED TO SOLVE PROBLEM**	**RAD APPROACH DESIGNED TO SOLVE PROBLEM**
Software Complexity	Training, Peer Reviews, Software Product Engineering	Prototyping versions of software
Ability to estimate cost, schedule, size of program	Risk Management, Configuration Management, Project Planning and Tracking	Development Tools, Time Boxing
Unstable requirements	Requirements Management, Configuration Management, Peer Reviews, Product Engineering, Intergroup Coordination	Prototyping versions of software
Poor problem solving/decision making	Project Planning and Tracking, Subcontract Management, Quality Assurance, Organization Process, Integrated Software Management, Requirements Management	High level of communication due to small team, Development Tools

Table 1. Processes Designed to Solve Software Development Problems

VI. SOFTWARE PROCESS IMPROVEMENT EFFORTS IN INDUSTRY

The two companies I chose to describe in this section are both successful and leaders in their area of expertise. One company uses a formal approach to software development and the other has, in the past, used a very informal approach to software development.

In July 1996, The Defense & Space Group, Boeing Space Transportation Systems (STS), achieved a Level 5 rating using the SEI CMM. This rating was the result found when a CMM-Based Appraisal for Internal Process Improvement (CBA IPI) was performed. This assessment took 10 days and required that evidence of institutionalization be provided for all software development activities. This significant achievement illustrates how customer interest has shifted from the technology focus of the 1960s and 1970s to the cost & schedule focus of the 1980s and finally to a process focus in the 1990s. Because the STS made process improvement a high priority, it demonstrated a high level of maturity in a formal assessment. [Ref. 10]

As stated in [Ref. 11], Boeing feels the most tremendous benefit of high-maturity processes has been the Air Force's satisfaction and confidence in their organization – not because of their CMM rating, but because of the high quality of their product and predictable cost and schedule. The Air Force is Boeing's major customer.

Another unexpected benefit of reaching a Level 5 was revealed when STS responded to an Air Force Software Development Capability Evaluation (SDCE) as part of a proposal preparation. Having CMM-based processes in place made it easier to respond to the difficult questions, and documentation supporting STS's claims was readily available for inclusion into the responses. [Ref. 11]

The practices used by Boeing's STS organization are described here briefly and in detail in [Ref. 12]. An internal Software Engineering Process Group (SEPG) managed the organizational processes. Processes were updated as needed, training was developed and implemented, and emphasis was placed on a standard software development process. The metrics developed and reported included defect rates, training effectiveness, defect probability and other trend measures. Inspections were performed often since Boeing thinks that inspections are the single most effective method of reducing defects. In their

application, defects were reduced by 85%. Cycle time was reduced up to 50% and productivity was increased by 240%. The cost-to-benefit ratio of 1:7 was realized along with cost underruns. With numbers such as these, it is clear that the CMM has helped Boeing STS to become successful in many ways.

In contrast, Microsoft, also a highly successful organization, has had a different approach to software development, which I will describe briefly here. Their approach is described in detail in [Ref. 13].

In the early days of Microsoft, the development process was more informal and there was less emphasis on schedule methodology and software architecture. Perhaps because of their lack of formal training, most of these developers did not follow the highly structured software development methodologies created by DoD and large corporate MIS departments. They often developed software without a formal specification or design.

Microsoft relied heavily on individual superstars to develop software. Although this strategy worked well for them for awhile, they also encountered many problems, e.g., the superstars were hard to find; the people who were hired to maintain the code written by the superstars had difficulty understanding their code; the superstars did not always understand what the market wanted; and if too many of them were put together on a team, design decisions became a real problem. They did not always work well together.

Because Microsoft was criticized in the mid-1980's for writing software which was technically excellent but difficult to understand and hard to use, Bill Gates began to hire marketing specialists to reorient Microsoft to focus on the customer. They also introduced a program management function to formalize design, coordinate product creation functions (development, testing, and user education), and perform support functions such as manual reviews, and competitive product evaluation. Teams of about a dozen software developers for each major project were established. Other major competitors of the time used much larger teams (often 100 or more). It is interesting to note that Microsoft's cost per line of code developed was significantly lower than the industry average, which was about $125 in 1989.

While Microsoft had become somewhat more formalized, the basic truth was that the developers still had ultimate control of the process and the code. The company had a deep underlying philosophy that people know what they are doing and will try to do the right thing.

In 1984, a team of highly experienced developers was established to develop Word for Windows. The project was scheduled for completion by October 1985. They had numerous problems and in July 1986, the project manager left the project. A new team was formed. They ended up throwing away everything that had been done for the last year and started over. They were a year behind schedule from the start. The new team was almost entirely staffed with new hires. Because this new team was under heavy pressure to deliver a system, they tended to do the minimum amount of work necessary on a feature. The end result was that the testing and debugging phase took much longer than expected. Staffing of the project changed due to illness/burnout of employees. Word for Windows was finally released on November 30, 1989.

In order to learn from the mistakes of previous projects, Microsoft had instituted a policy of reviewing every project upon its completion. Statistics were collected into a document along with minutes of meetings held during the project. This document (called the postmortem) was distributed to all managers in the business group and to senior managers. The postmortem for this project was four times larger than the average project postmortem. While [Ref. 10] does not describe what Microsoft actually did as a result of the postmortem, it does describe some of the ideas for improving product development that were discussed in the postmortem. The majority of developers enjoyed the informal approach to software development, however, some felt that a more structured development process would substantially improve the company's development performance. This would include relying on more formal project phases and strict milestones, as well as the implementation of formal structured methodologies for software development. Others felt the cause of Microsoft's problems was in its approach to project management. The current approach lacked focus and control. The developers felt that the program managers simply did not have the level of technical knowledge required to really understand a software development project. Other managers thought

the lack of a uniform strategy at the business level was the fundamental problem. They focused on the need for coherence across similar applications.

It is interesting to see that the problems and successes in industry are similar to the problems and successes experienced in government. The rest of the thesis focuses on the SPI efforts at SSC-SD and, in particular, one project at SSC-SD.

VII. SOFTWARE PROCESS IMPROVEMENT EFFORT AT SSC-SD

Software is a major part of many systems that SSC-SD acquires, develops, or maintains. Consistent management and engineering processes must be applied to the acquisition, development, and maintenance of software to identify and control the risks associated with software-intensive systems. In the current environment of rapidly changing technology and competition, the SSC-SD software engineering mission is to keep competitive by being a DoD leader in developing and acquiring quality software-intensive systems. [Ref. 14]

The Software Engineering Process Office (SEPO) was established as a result of an SEI assessment in 1988 and was tasked to assist the SSC-SD software community in identifying, documenting, institutionalizing, and improving the processes they use to develop and maintain their software products. As projects improve their processes, it is expected that the government will save dollars on contracts and in-house labor which is necessary in this time of shrinking budgets and downsizing edicts. The government contractors, on the other hand, can expect to see an increase in productivity which will give them a more competitive edge in bidding for contracts and the company's capacity to do work will increase thereby increasing profits. SEPO has been developing policies and processes for project managers to use as well as providing in-house training. The SPI effort at SSC-SD uses CMM as a guide for improving organizations' software engineering process. The rest of the thesis describes the costs and benefits of the SPI activities implemented on an SSC-SD project called SmartNet.

VIII. DESCRIPTION OF THE SMARTNET PROJECT

SmartNet is a project that has been working to improve its software development processes for the last three years. Prior to December 1994, several years were spent by the project manager doing the necessary research in high performance and distributed computing needed to get SmartNet started. In December 1994, SAIC joined the team and started implementing SPI activities. It is fielded to several users. The technical team currently fixes problems, adds enhancements as approved, and distributes new software releases periodically. The latest software version contains 92,345 lines of code. This paper will describe the processes they have chosen to implement and the costs and benefits associated with the activities performed. This section describes the project and a little bit about the history and the people working on it.

SmartNet is a scheduling tool for distributed High Performance Computing (HPC). SmartNet's goals are to:

- Maximize computing power
- Increase throughput
- Optimize cost-effectiveness
- Leverage existing resources
- Ensure robust scheduling

SmartNet provides an HPC scheduling framework for near-optimal and simultaneous employment of up to hundreds of resources (heterogeneous and homogeneous, local and remote) for thousands of tasks (heterogeneous and homogeneous). Performance improvement of over one order of magnitude decrease in computing time with existing tasks on existing resources has been achieved and documented.

SmartNet determines scheduling decisions by matching compute characteristics (code/algorithms and data) to resources (processors and networks) which are collectively called a Virtual Heterogeneous Machine (VHM). [Ref. 15]

SmartNet can be beneficial to any system that uses multiple resources. It has to be integrated into current systems. Some of the current users of SmartNet are listed and their applications described in this paragraph. NASA Earth Observation System (EOS) at

Goddard Space Flight Center in Maryland extracts data from satellites and they use SmartNet to plan what processes go to what computers. The National Institute of Health (NIH) wants to use resources after hours. SmartNet has helped them to distribute jobs. Joint Task Force Advanced Technology Demonstration (JTF ATD) project has integrated SmartNet with its model server for the Pacific Disaster Center. They run simulated disaster models and SmartNet will act as a scheduling advisor within different sites. Florida State University, National Security Agency (NSA), Navy Simulation Systems (NSS), and the USS Coronado are among the numerous test and evaluation users.

The Heterogeneous Computing Team (HCT), led by Richard Freund, consists of SSC-SD personnel in Code D4223, SAIC personnel, consultants, and academia from Naval Post Graduate School, George Mason University (GMU), University of Cincinnati, and Purdue. The SmartNet product is developed at SSC-SD and is funded by Defense Advanced Research Project Agency (DARPA) Information System Office (ISO), Deputy Under Secretary of Defense for Space Integration (DUSD (SI)), NASA EOS, NSA, High Performance Computing Modernization Office (HPCMO), and Space and Naval Warfare Command Center (SPAWAR).

The team consists of twelve full-time and seven part-time people plus various academic collaborators. They have three full-time and two part-time managers; eight full-time and two part-time people that perform research, programming and engineering support; one full-time and one part-time administrative assistant; and two part-time consultants.

Richard Freund, the project manager, has stated that his software process improvement goals and project goals are inter-related. They are:

- To achieve a CMM maturity level 3
- To produce a high quality product with high reliability while maintaining a high level of control in configuration management.
- To successfully market the SmartNet product

In order to develop and maintain a high quality product that is highly reliable and controlled, they must have processes in place and in practice to consistently achieve this goal. These factors are mandatory to successfully market SmartNet. This project was

evaluated by an internal SAIC Common Approach Based Appraisal (CBA) Internal Process Improvement (IPI) assessment team and was found to satisfy all activities of a CMM level 3 project. SmartNet draws on both SSC-SD Program Management and SAIC Management and process improvement activities defined and implemented by SAIC to satisfy the organizational support requirements for that level. Since the processes used on SmartNet are SAIC processes (versus SSC-SD processes developed by the SEPO office), the "organizational" requirement/definition was not met. If SmartNet were to implement SSC-SD SPI processes, they could be re-evaluated for a level 3. A formal independent evaluation is planned in the fall of 1998; SmartNet was going to be one of the projects appraised at that time. Unfortunately, the project manager is leaving SSC-SD and the project is coming to an end in December 1997.

SAIC's SPI goals are to achieve a CMM maturity level 3 in order to increase quality and quantity of work performed as well as continue to be competitive for government contracts. They achieved their goal.

IX. SPI ACTIVITIES IMPLEMENTED ON SMARTNET

According to the SmartNet configuration manager, all SEI-CMM level 2 and 3 KPAs are addressed on the SmartNet project. All SEI-CMM level 2 and 3 activities are defined and practiced in an effort to solve the problems discussed earlier in the thesis. The basis for the processes used by SAIC is described in several volumes of a document called *Common Approach to Software Development and Maintenance.* This document is described below.

Volume 1. *Policy and Processes* contains the following sections:

- Software Development and Maintenance Policy
- Software Development and Maintenance Process
- Software Management Process
- Software Review Process
- Software Test Process
- Software Quality Assurance Process
- Software Configuration Management Process

Volume II. *SEI CMM Compliance Matrix*

Volume III. *Implementation and Tailoring Matrices*

Volume IV. *Sample Forms, Checklists and References*

Volume V. *SAIC Metrics Handbook and Collection Guide*

- *SAIC Handbook for Software Measurement Reporting Formats*

Volume VI. *Training Program Definitions and Guidance.*

SAIC also has a Quality Program, a Software Training Program and a Measurement Program. Employees are required to learn and practice the methods described in the Common Approach.

The SmartNet team has implemented an Oracle database to track and store their Software Change Requests. They keep metrics on the actions taken by the Software Configuration Control Board (SCCB) in this database. Reports are generated from this database and are included as Appendices to this document. They also keep hardcopy files of SCRs. These files and the database are checked/audited every three months for

accuracy. CVS is used for version control. They are currently in the process of automating their processes. The following paragraphs describe the reports.

The *SmartNet - Software Change Request* is filled in on-line by users/testers of SmartNet whenever a problem has been found or an enhancement has been identified. There are fields identified to store the following information: type of SCR, date of SCR, status, priority, deadline, level of effort expected, software lines of code (SLOC) affected, where the problem was found (design review/code review), the assignee, name of submitter and email address, version of software, tests run, and description of the problem/enhancement. When the SCCB meets, the new SCRs are discussed and action is assigned and noted on the SCR. A copy of this report is found in Appendix A.

The *Actual vs. Estimated Level of Effort (LOE) Report* is used to track the amount of time estimated to fix the change and the actual amount of time it took to fix the change. It can be sorted by SCR number or by assignee. This report covers SCRs and Action Items (AI). The difference between the estimated LOE and the actual LOE is calculated and written on the report. If the difference is a positive number, the assignee fixed the problem faster than expected. If the difference is a negative number, the assignee spent more time than expected fixing the SCR. On the report sorted by assignee, a summary is provided for each assignee of the average time spent by the assignee on each SCR and a summation of the total number of hours spent fixing the SCRs assigned to them. The summary also includes a summation of the estimated minus the actual hours to fix the SCRs and an average of that number over all the SCRs. On the report sorted by SCR number, a summary is also provided at the end of the report. The summary includes the total number of hours estimated and actually spent fixing all the SCRs in the system and the difference between the two. The average of the actual, estimated and difference is also calculated. A copy of this report is found in Appendix B.

The *Closed AIs/SCRs Report* can also be sorted by SCR number or by assignee. This report lists each SCR and AI along with the deadline for fixing the SCR or completing the AI. The actual LOE is also listed on the report. On the report sorted by assignee, a summary is provided for each assignee of the total number of hours spent

fixing SCRs/AIs and the average number of hours spent. A copy of this report is found in Appendix C.

The *SCRs/AIs Closed Between <start date> and <end date>* lists all the SCRs/AIs that were closed within the dates specified in the starting date and ending date (i.e., SCRs/AIs closed between 01-Mar -1997 and 05-Sep-1997). The SCRs are sorted by date closed. This report can also be generated for SCRs/AIs opened between specified dates. A copy of this report is in Appendix D.

The *Team Power Report* gives a level of effort analysis and current status of each open SCR/AI. This report is reviewed each week at the Configuration Control Board meeting and provides the project manager with a summary of what the team is working on. The report is sorted by assignee. It lists each SCR/AI assigned to the engineer, the due date of the SCR/AI (Q1 through Q4 for which quarter of the year), and the Percentage of Effort (POE) set by the task leader. It also lists the priority of the SCR/AI (normal, normal+, urgent), the percent done, the estimated level of effort (LOE), the current LOE, and the adjusted LOE. A rule of thumb used to determine the percent of work done is described below:

- the first 25% of the software developers time will be spent defining requirements
- the second 25% of the work is designing the change
- the third 25% is coding the change
- the last 25% is testing the change

So, if the software developer was currently in the process of coding the change, it would be estimated that the percent done is 50%. The estimated level of effort is the number of hours expected to be spent fixing the SCR/AI. The current LOE is the cumulative number of hours actually performed. The adjusted LOE is the number of hours expected to be spent based on percentage done and the current LOE. For each assignee, a summary is provided that lists the hours remaining, the total LOE, the LOE balance, and the adjusted LOE balance. The hours remaining is the amount of hours left to fix the SCR; it is the difference between the due date of the SCR and the current date. The total

LOE is the sum of the LOE of all SCRs/AIs assigned to each person. The LOE balance is calculated by the formula

hours remaining * POE - estimated LOE.

The adjusted LOE balance takes into account the percentage done. A copy of this report is found in Appendix E.

The *SCR Metrics for Problem SCRs* contains several "mini reports" based on metrics entered into the database. One report is called *SCR Average Age Report.* The data in these report covers a specific time frame (in days) entered by the operator. It lists all SCRs opened as of the end of each month specified by the time frame entered, and identifies how many of those SCRs have a critical/urgent priority. The data in this report is skewed due to the fact that low priority SCRs with a due date of several months or years away are included in the data along with high priority SCRs with early due dates. Another report is the *SCR Open Activity Report.* This report lists the number of SCRs that have a priority of critical, urgent, normal, and low as of the end of each month specified by the time frame entered. The *SCR Open/Close Rate Report lists* the number of SCRs opened and closed at the end of each month specified by the time frame entered. The *SCR Total Open Report* lists the number of all SCRs opened as of the end of each month specified by the time frame entered. This report also lists the number of critical/urgent SCRs. Another set of these "mini reports" can be generated for SCR problems and enhancements. This report is called the *SCR Metrics for Problem and Enhancement SCRs.* These reports are found in Appendix F.

The SmartNet team has implemented a "Home Team Page" on the web to communicate among all of the developers and consultants and exchange technical ideas as well as meeting information. They call this "Newsgroup" discussion.

Each release of a new software build requires 2-4 weeks of testing prior to release to the customers. The developers usually write the tests while others, less familiar with SmartNet, actually run the tests. SCRs are written by the test team and fixed by the developers. The documentation is also updated to reflect the software changes implemented.

Each build is documented in a Version Description Document (VDD) that lists the lockdown and release dates, the status of each SCR included in this version, the requirements for the version and the lower priority fixes included. This information is stored in a repository. An example of the VDD is in Appendix G.

SAIC provided training for its employees. The costs of this training are discussed in the next section. Classes taken are:

- Common Approach Orientation (3 hours)
- Software Project Management (24 hours)
- Requirements Management (24 hours)
- Configuration Management (8 hours)
- Introduction to Software Quality Assurance Course (4 hours)
- Software Quality Assurance for Practitioners (16 hours)
- Estimation (4 hours)
- Peer Reviews (8 hours)
- Testing (8 hours)
- Engineering Principles (8 hours)
- Metrics (8 hours)
- Process Group Operations (4 hours)
- Common Approach Based Appraisal-Internal Process Improvement (CBA-IPI) (30 hours)

SSC-SD also provided training for their employees on the SmartNet team. Some of the classes taken were:

- Software Project Management Course (40 hours)
- Grammar Brush-up for Writers (8 hours)
- Contemporary Navy Writing (16 hours)
- Giving Technical Presentations (16 hours)
- Effective Leadership for Women (8 hours)

One of the activities the SmartNet team implemented is the peer review. They have three types of peer reviews: formal design reviews, formal code reviews, and informal one-to-one reviews. The majority of the reviews done on SmartNet were design

reviews. As of December 1996, the team had only done a few code reviews. The design reviews are the time to focus on clarifying the requirements, discussing design issues and planning the design of the project. They did not keep records of defects found in design reviews because the purpose of the design review was to work out differences or misconceptions, not to generate Software Change Requests. The work surrounding the review topic was covered under an already open SCR (since they address bugs and enhancements under the same tracking system). They do keep track of "where" a defect is found in the SCR database, but that information is used mostly during the test phase. They do not keep track of defects found in the informal one-to-one review.

The software development team uses a Software Detailed Design Review form and a checklist to help analyze proposed design changes. A review panel consisting of the Software Manager, the Systems Engineer, and other software developers/engineers on the team discuss the design changes. At the design review meeting, the assignee must present his/her design and implementation plan. Each engineer who attends the design review must fill out the design review checklist. The software manager fills out the design review form indicating acceptance or rejection of the design change. If the change is accepted, the assignee must submit a thoroughly documented design in paper.

X. ANALYSIS OF COSTS OF SPI ACTIVITIES IMPLEMENTED

The tangible costs of implementing and maintaining SPI activities are evidenced in time spent. Table 2 is a summary of estimated time spent in this effort. The time included:

- training
- setting up and maintaining the SmartNet database
- SCCB meetings
- risk assessment meetings
- quality assurance audits
- process audits
- managing SCRs requirements management
- tracking time spent
- reporting
- unit and/or component test
- managing the source code itself
- periodic reviews

It is impossible to quantitatively compare these efforts to the activities that existed prior to the implementation of formal process (e.g., planning, mitigating risk, addressing quality issues, evaluating/improving productivity, and reviewing the product) because there are no metrics on the former informal processes used by the SmartNet team. No software costs for the ORACLE database are included because SSC-SD already had purchased the software.

The training costs for SAIC includes learning about CMM, how to implement CMM activities, and how to assess the maturity level of projects. The cost of designing and giving the courses is not included in this estimate since SAIC gives these classes to employees on all projects within the organization, not just SmartNet. The costs include labor hours spent in training. The dollar figure was derived from the number of class hours multiplied by the number of attendees multiplied by hourly rate. The hourly rate varies for each category of labor. The categories on SmartNet are project managers, project leads/senior engineers, and software engineers.

SSC-SD also provides training for its employees. The specific classes are listed in the previous section of this thesis. The total costs were derived from the number of people attending the class multiplied by the number of hours plus the tuition costs for each student.

An intangible cost described by SAIC personnel working on SmartNet is the time and effort expended to change the culture of the organization. The people on the project did not want to become involved with the SPI efforts and, as a consequence, were "fighting" the system. The employees wasted time as a result of attitudes against SPI. The solution they found was to make the SPI policies flexible enough so that the practitioners could use them. It is interesting to note that according to one employee, the same people who were resistant to implementing SPI activities in the beginning would not want to work on a project that didn't practice SPI now.

Description of Costs	Costs incurred by SSC-SD	Costs incurred by SAIC	Total
Training Costs (see text for details)	$ 28,960	$ 32,654	$ 61,614
ORACLE Database Set-up (estimated DP-3 half workyear FY95)	$ 68,200		$ 68,200
Action Items Implemented (estimated 600 hours @$41/hr)		$ 24,600	$ 24,600
Database Maintenance (estimated DP-3 1 day/wk for 3 years)	$112,066		$112,066
Configuration Management (estimated half workyear @ $41/hr)		$ 36,080	$ 36,080
Developers (estimated 1 day/wk for SCCB/peer reviews/metrics @ $41/hr)		$ 14,432	$ 14,432
Total	$209,226	$107,766	$316,992

Table 2. Summary of Costs Incurred Implementing SPI Activities

XI. ANALYSIS OF BENEFITS OF SPI ACTIVITIES IMPLEMENTED

The SmartNet project manager, Richard Freund, believes that it takes two to three years to see the benefits of SPI efforts. The SPI activities on this project started three years ago. One of the benefits he has seen is that he has better management control over his program. The mechanisms that the team has in place to accomplish this are: the informal design/peer reviews, the website "Newsgroup", the maintenance of the SmartNet database, the generation of reports that give the status of the software development effort, and the weekly SCCB meetings. With better management control, Richard has more time to market SmartNet and build business.

As described by the software development team, some of the tangible software quality improvements on this project are fewer system failures, better overall performance, and delivery of software builds closer to schedule. Table 3 lists the data available on their software delivery schedules.

The team also thinks that the software design is well documented and understood by the developers. This enables developers to fix SCRs or add enhancements with less effort required than in the beginning of the project when the software design was not as well documented. Figure 1 shows the estimated and actual LOE expended on SCRs for the project. The difference between the estimated and actual LOE is calculated by subtracting the actual LOE from the estimated LOE. The result is included in Figure 1. In most cases, the actual LOE exceeded the estimated LOE. This fact adds testimony to the difficulty of estimating software development even with SPI processes in place. The peaks of activity on this project occurred in preparation of the release of versions 2.1-beta, 2.5, and 2.7 of the software. Version 2.1-beta was the first documented software build released on 15 February 1995. Their SPI effort started in December 1994. Version 2.5 was released on 5 February 1996. According to the reports generated from the database, the previous version (v2.3) was released almost 9 months prior to v2.5. So, it seems reasonable to expect the most amount of activity happened during the longest time between software versions. The average length of time between software versions was 3.86 months. The third peak of activity occurred in preparation for release of version 2.7 in October 1996. Figure 2 shows the estimated costs associated with the actual LOE

expended fixing SCRs. This was calculated by multiplying the LOE and the current rate of a software engineer.

Software Version	Planned Release Date	Actual Release Date
Version 2.1-beta		15 February 1995
Version 2.1		21 February 1995
Version 2.2a		4 May 1995
Version 2.3		25 May 1995
Version 2.5		5 February 1996
Version 2.6	29 March 1996	22 April 1996
Version 2.7	01 October 1996	10 October 1996
Version 2.7.2	05 March 1997	05 March 1997
Version 3.0	01 October 1997	Still unreleased

Table 3. SmartNet Software Release Dates

Figure 3 shows the number of SCRs closed and opened for each version of software released. The data used for this chart is found in Appendix F. This chart also shows the number of critical and urgent priority SCRs opened during the time frame in between each software version. The highest number of SCRs opened and closed occurs at the same time version 2.5 is released. As mentioned previously, the length of time

between version 2.3 and version 2.5 was more than twice as long as the average time between versions, so a higher number of SCRs would not be a surprise. The number of SCRs opened and closed continues to diminish for the subsequent software versions even though the time in between versions is longer than average.

According to the project management team, some of the intangible benefits are customer satisfaction due to delivering a higher quality product that is built to customer specification, providing up-to-date user documentation, and conducting effective (crash free) demonstrations. The number of SCRs written has diminished by approximately 45% over the last year and a half, which could indicate that SmartNet software is of higher quality than in years past.

Another indicator of higher quality is illustrated in Table 4. It shows the estimated vs. actual SCRs written along with the reduction in defect rates experienced by this project. According to [Ref. 16], many SPI methods reduce the number of defects induced into a product, thus reducing rework costs. Defect rates are measured in terms of "Average Defects per KSLOC." Several sources quoted in [Ref. 16] have shown that 7 defects per KSLOC is a typical defect rate throughout the development process for new code. Estimated SCRs is calculated by multiplying 7 times the KLOC. As evidenced in Table 4, SmartNet's defect rate is much lower. SAIC provided the SLOC for versions 2.7.2 and 3.0. I used the same SLOC as version 2.7.2 for all other versions of software. Since most SCRs were found and written during the testing phase for each version, a possible conclusion is that the reduction in defect rates is the result of informal peer reviews and design reviews.

The SAIC employees are not working the amount of overtime they did in the beginning of the project. However, according to the software manager, although some of the peaks of overtime have disappeared since they started doing better process management, the overall amount of overtime has probably stayed level because the amount of work they attempt to do seems to expand to fit the time available. Other benefits are that the team is better educated due to extensive training offered at SAIC and SSC-SD. The people on the project are more accountable for their work and feel a sense of responsibility to make sure their work is done well. The employee morale is also

higher. This can be evidenced in the fact the members of the team participate in meetings by providing ideas for improving their product technically and ideas on improving marketing efforts. Communication is better between team members. They seem to have better attitudes and would not want to work on projects that did not have SPI activities implemented.

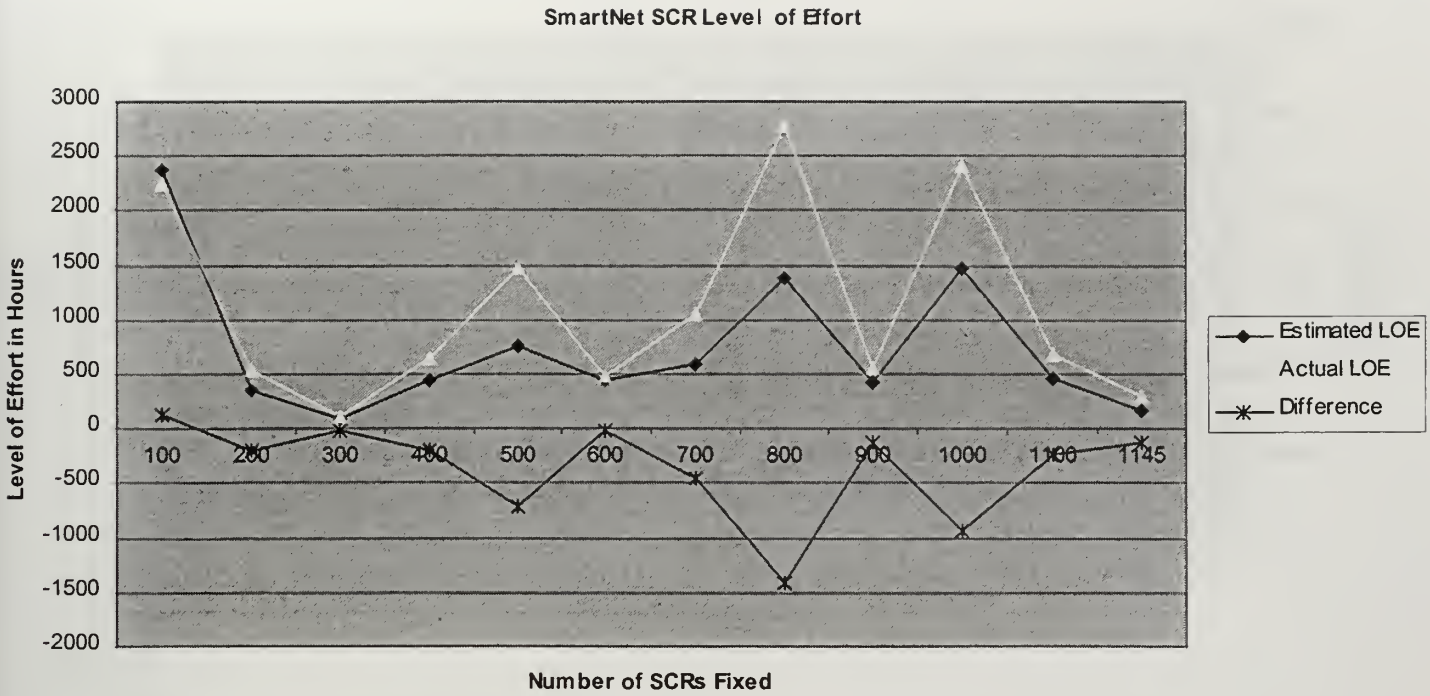

Figure 1. SmartNet SCR Level of Effort

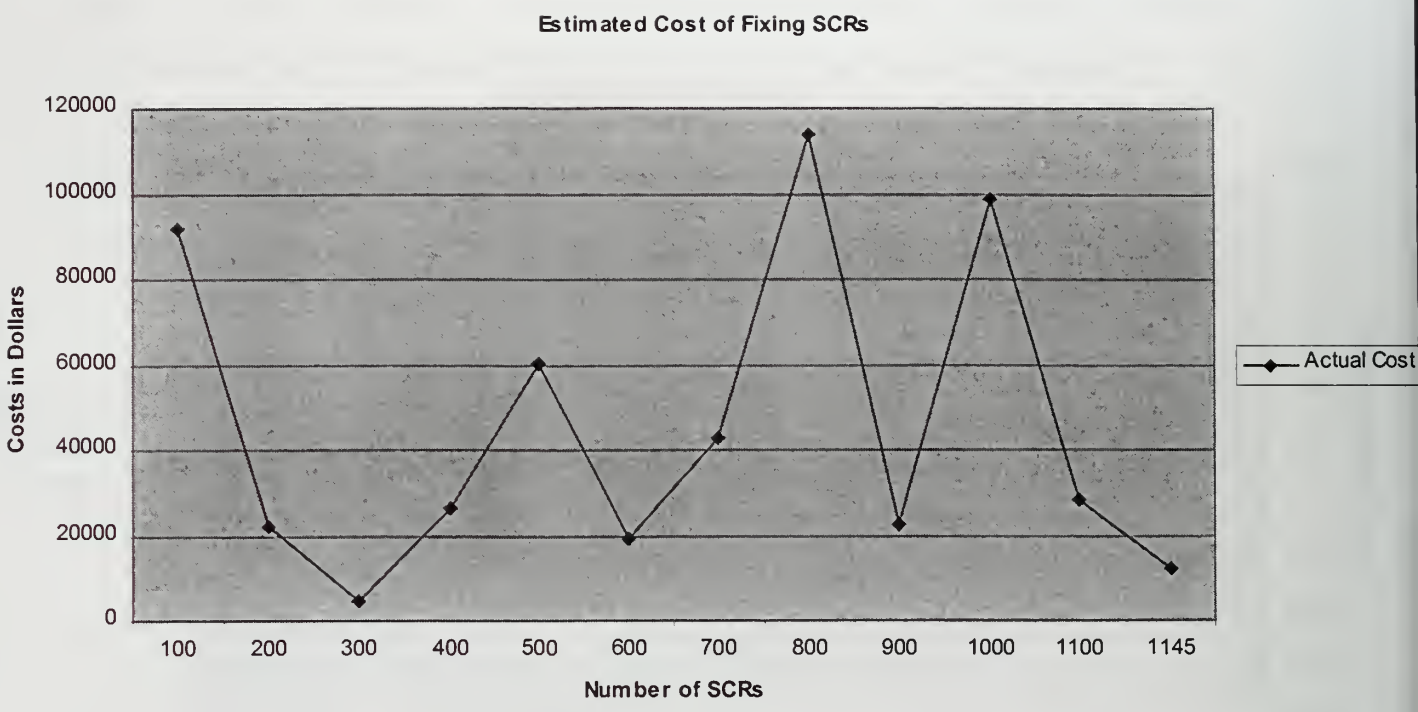

Figure 2. Estimated Costs of Fixing SCRs

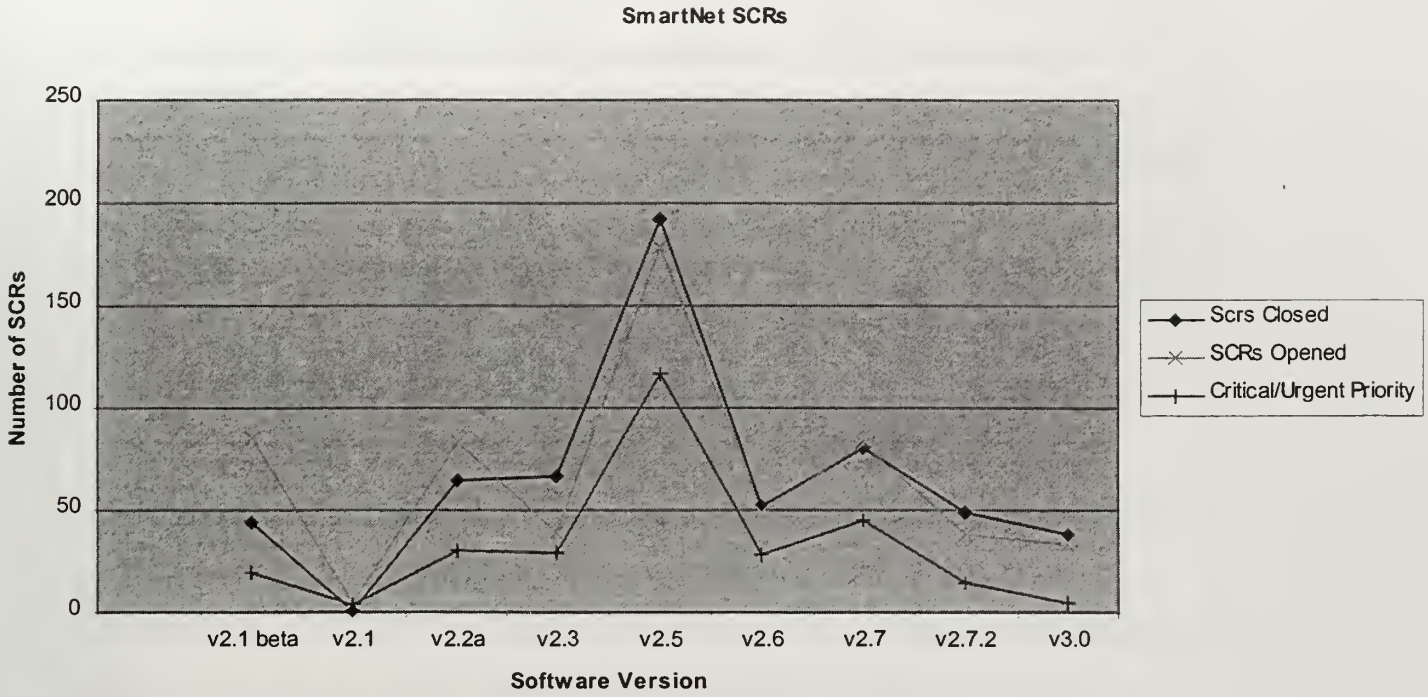

Figure 3. SmartNet SCRs

Software Version	KLOC	Estimated SCRs	Actual SCRs Opened	Reduction in Defect Rates	Cumu-lative # SCRs	Cumu-lative Reduction in Defect Rates
V2.1 beta	93.603	655	85	87%	85	87%
V2.1	93.603	655	5	99%	90	86%
V2.2a	93.603	655	83	87%	173	74%
V2.3	93.603	655	39	94%	212	68%
V2.5	93.603	655	178	73%	390	41%
V2.6	93.603	655	49	93%	439	33%
V2.7	93.603	655	82	87%	521	20%
V2.7.2	93.603	655	38	94%	559	15%
V3.0	85.951	601	33	95%	592	2%

Table 4. Defect Rates of SmartNet Software Versions

XII. CONCLUSIONS

According to the SmartNet Project Manager and a few members of the development team, the SPI activities that resulted in the most benefits are in the KPA of Project Tracking and Oversight. The project manager and software manager have more insight into the status of the software development effort, and therefore can make better decisions. The SmartNet team repeatedly praised the process used to track SCRs as an effective and efficient process. This was noted at the CBA-IPI conducted in May 1996. The reports generated from this SCR database provide the knowledge needed by management to better control the project and make sound business decisions.

To summarize, the primary costs of implementing SPI activities on SmartNet were the costs to train the employees; setup and maintain the SCR database; and participate in SPI activities, including SCCB meetings, risk assessment meetings, quality assurance audits, requirements management, keeping metrics, reporting, testing, and periodic reviews.

The benefits resulting from their efforts are:

- Better management control over the project
- Better overall performance of the software
- Better documentation
- Software delivery closer to scheduled date
- Higher quality software
- Higher customer satisfaction
- Improved employee morale
- Better communication among the team
- Less overtime required to get the job done
- Employees better educated
- Increased employee pride in their work and increased responsibility and accountability for their work.

In comparison, the benefits received from the efforts made on SmartNet are similar to what was documented in *A Business Case for Software Process Improvement.* [Ref. 16] One report sited states the many non-measurable benefits from a SPI program

include improved morale by the developers, increased respect for software from organizations external to software and less required overtime. Other benefits from SPI described in this report are that companies feel they are more competitive, improved customer satisfaction, and more repeat business from their customers. Another report claims that the first benefit resulting from SPI is the ability to meet schedule. Based on Raytheon's SPI effort, benefits are that employees feel the company wants them to do a good job, higher employee morale, less absenteeism, lower attrition rates, and fewer nights and weekends required by employees. Most of these secondary benefits are hard to quantify and hard to measure. It is encouraging that SmartNet experienced the same benefits as other companies and projects that have implemented SPI activities.

As documented in [Ref. 17], the benefit most frequently noted by the research participants concerned attitudinal changes. The morale and confidence of the developers improved significantly. Participants also attributed less overtime, less employee turnover, improved competitive advantage, and increased cooperation between functional groups as benefits resulting from process improvement initiatives. SmartNet experienced the same benefits.

In summary, this thesis has discussed the issues and problems involved in software development and described both a formal and an informal approach to solving the problems inherent in software development. The discussions of SPI efforts in industry and government and the case study give additional credence to the idea that a well-defined process (whether formal or informal) will help any software development project to succeed.

APPENDIX A. SOFTWARE CHANGE REQUEST

SCR# 1063 SmartNet - Software Change Request
SCR# 1063

Title: SNAP time clock runs slow in LIVE mode

Type: PROBLEM Post-Mortem: N Submit Date: 18-OCT-1996

Status	Priority	Deadline	LOE	SLOC	DR	CR	Assignee
CLOSED	URGENT	Q3T-1996	5	0			Bill Adsit

SCCB Action: _____ _____ _____ _____
SM VCM SQA Date

Resolution: _____ _____
Closing Engineer Date

Submitter: Bill Adsit
E-Mail: adsit@nosc.mil
Version: 2.7
Tests: ap014

Description:
This problem showed up on ap014, step 06 and 07. It gets increasingly worse the longer sn-ap runs. The problem is that the time field is updating slower than real time. This "current time" value is used in SNAP to calculate ETC, running time, and other things. Some of the symptoms are: jobs that don't start un-filling right away, jobs are "done" before they finish un-filling (short jobs may even disappear before they start unfilling), and when the error is over 10 or 15 seconds (after running SNAP for as littleas five minutes), some of the Done jobs may show up in the Scheduled Jobs window. I looked at the code and determined that the problem is SNAP's use of PlayrateTimerCB in Live mode. This resets a one-second timer every second. Of course, it takes a few microseconds to respond to the timeout and restart the timer; Thus, every second, "current_time" gets a few more microseconds behind. The solution is to use REAL-TIME when running in Live mode. The libmrh Virtual time class method (i.e. current_time = VirtualTime::getTime()) should be used in the ActionWin::set_time() method when in Live mode. This would update the "current_time" variable at least once per second.

APPENDIX B. ACTUAL VS. ESTIMATED LEVEL OF EFFORT (LOE) REPORT

A c t u a l vs. E s t i m a t e d L O E R e p o r t
(Version 2)

September 5, 1997

Assignee	SCR#	AI#	Title	Actl. LOE	Est. LOE	Est-Actl
Bill Adsit	545		Runner locked up X display when trying to cancel jobs	4	8	4
	824		Problem modifying fields in the Editor on Solaris 2	6	6	
	875		Modify SNAP to handle job abort messages	2	8	6
	885		Grand Unified SNAP Enhancement List	204	80	-124
	894		Snap background display	1	8	8
	895		Snap Locks up when stepforward button is pressed.	1	8	8
	896		Clicking caused display to disappear with Segmentation fault (core dum		8	8
	899		Machine names invisible in HP vue environment	5	8	3
	902		Fast forward speed lingers while playing	2	8	6
	903		SNAP needs feedback when logfiles don't load	14	4	-10
	922		SNAP Visual enhancements	95	40	-55
	968		Implement selection of scheduler in runner gui	47	16	-31

SmartNet Team Report
PAGE: 1

Actual vs. Estimated LOE Report
(Version 2)

September 5, 1997

Assignee	SCR#	AI#	Title	LOE	Actl. LOE	Est. Est-Actl
Bill Adsit	1019		sn-ap color map problems	1	4	3
	1022		Minor SNAP Visual Enhancements	20	8	-12
	1024		Big job kills sn-ap	3	3	
	1034		Testing Additions: Need a test for each GUI re. XUSERFILESEARCHPATH en	4	4	
	1041		ETC value within sn-ap is incorrect	10	4	-6
	1055		SNAP needs to limit display of *LONG* jobs/schedules and/or limit disp	7	8	1
	1059		SNAP error with 3-digit hours in ETC field	3	4	1
	1060		SNAP still has memory problems creating pixmaps on some X-Terminals	5	5	
	1063		SNAP time clock runs slow in LIVE mode	5	5	
	1073		SNAP legend window needs close button and Windows menu needs checkboxe	34	8	-26
**********				-----		--------
avg				20		-9
sum				511		-224

SmartNet Team Report
PAGE: 2

APPENDIX C. CLOSED AIs/SCRs REPORT

C l o s e d A I s / S C R s R e p o r t
(Version 1)

September 8, 1997

SCR#	AI#	Title	Dline	Actl LOE	Assignee
1		Global Override	IPPS	16	Matt Kussow
2		Database Editor Display	JTF	27	Dave Schwarze
3		SmartNet Runner crash	JTF	3	Wanda Lam
4		Editor crash in pop-up menu	1995-Q2	8	Dave Schwarze
5		SmartNet halt due to Runner.	JTF	28	Mark Campbell
6		Schedule Monitor Display lag	JTF	18	Steve Ambrosius
7		Schedule Monitor problem	JTF	45	Dave Schwarze
8		Network Data editing	IPPS		Dave Schwarze
12		SmartNet schedule-monitor crash	IPPS	9	Dave Schwarze
13		Determine need for FOM class and get rid of it if possible.	JTF	80	Matt Kussow
14		SN BUG REPORT	JTF	1	John Lima
15		SN UPDATE REPORT	JTF	1	Brad Rust
16		Two 'exper_adds' occur for one job(should only be one)	JTF	1	Dave Schwarze
17		Newer version of libmrh desired for SN	IPPS	8	Brad Rust
18		Replacement of (GNU) String class with MString class	IPPS	80	John Lima
19		RUNNER modification	JTF	1	Wanda Lam
20		Runner "Clear Schedule" not successfully canceling jobs.	1995-Q3	1	Mark Campbell
21		Rescheduling too often.	JTF	2	Elaine Keith
22		Editing Global Overrides.	IPPS	1	John Lima
23		Saving Argset information in Runner doesn't work properly.	IPPS	6	Wanda Lam
24		Runner's "Save Argset" should use File Selection Wiget	JTF	8	Wanda Lam
25		Improve security robustness of Message Object classes.	1995-Q3	30	Mike Halderman

SmartNet Team Report
PAGE: 1

APPENDIX D. SCRs/AIs CLOSED BETWEEN <START/END DATE>

S C R s / A I s C l o s e d
Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:47:26 PDT 1997

SCR/AI #	SCR Title	DATE_CLOSED	Assignee
SCR #1091	make a conversion routine for "old" log file formats.	07-MAR-1997	Marc Weissman
SCR #1103	Minor modifications for NASA planning	07-MAR-1997	Matt Kussow
SCR #1104	HP CC -O compiler crash on utils.cc	07-MAR-1997	Matt Kussow
SCR #1108	Job network profile information not getting to scheduler.	07-MAR-1997	Matt Kussow
SCR #1100	sn-ap needs to handle jobs that are already running.	07-MAR-1997	Marc Weissman
SCR #1089	Runner should start snap instead of monitor.	17-MAR-1997	Wanda Lam
SCR #1111	test ap001 failed	17-MAR-1997	Marc Weissman
SCR #1114	correct improper for statement In sn/server/src/Model.cc	17-MAR-1997	
SCR #1116	Minor modifications for NASA planning II	17-MAR-1997	
SCR #1115	take libdebug stuff out of libmrh	17-MAR-1997	
SCR #1113	setup new style SN_VERSION for main devel, and v2_7_a branch	17-MAR-1997	
SCR #1110	sn-submit006 step 2 failed to complete successfully on any platform.	17-MAR-1997	John Lima
SCR #1120	reconfigure version string to FMT: MAJOR.minor.build	03-APR-1997	John Lima

SmartNet Team Report
PAGE: 1

S C R s / A I s C l o s e d

Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:47:26 PDT 1997

SCR/AI #	SCR Title	DATE_CLOSED	Assignee
SCR #552	Secondary testing failed on test sn-control001	30-APR-1997	Brad Rust
SCR #1128	sn-submit fails to use machineLogin.db	30-APR-1997	Matt Kussow
SCR #1132	Constraint's scheduler combine Load enhancements	30-APR-1997	Elaine Keith
SCR #1095	RTUtils::FindRsh needs /usr/bin/rsh	30-APR-1997	Matt Kussow
SCR #1098	SmartNet Log File and Reader headers are allocated whenever included.	30-APR-1997	Marc Weissman
AI #56	Incorporate NASA data into SmartNet database	30-APR-1997	Matt Kussow
SCR #1070	sn-submit002, step01 failed on guava	30-APR-1997	Wanda Lam
SCR #1011	sn-submit does not return to command line prompt after completion	30-APR-1997	Matt Kussow
SCR #1125	SmartNet Runner Core Dumps When killing jobs during Runner003 Test on	29-MAY-1997	Matt Kussow
SCR #1109	Too much extraneous SNAP output.	29-MAY-1997	Marc Weissman
SCR #1124	ETC not calculated correctly with runner	29-MAY-1997	Matt Kussow
SCR #1003	SmartNet-Runner Taking Longer than 2 seconds to load an argument set	29-MAY-1997	Matt Kussow
SCR #1121	runner001 step 5 failed on all 3 platforms.	12-JUN-1997	Wanda Lam

SmartNet Team Report
PAGE: 2

S C R s / A I s C l o s e d

Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:47:26 PDT 1997

SCR/AI #	SCR Title	DATE_CLOSED	Assignee
	minimum size too small for font.		
SCR #1071	asynccomm010 failed on SunOS	01-JUL-1997	Terry Koyama
SCR #1081	Editor001 Test COMMIT Button Does Not Ungray	01-JUL-1997	Terry Koyama
SCR #1140	"sn-ap -d All -s -f new.snl -play" produces file name error.	01-JUL-1997	Terry Koyama
SCR #1138	Verbose is suppose to use cout (rather than cerr) as its output stream	01-JUL-1997	Terry Koyama
SCR #1136	Replace all cerr's, cout's and other output statements with debug	01-JUL-1997	Terry Koyama
SCR #1069	Editor001 Test Problems	01-JUL-1997	Terry Koyama
SCR #1049	Smartnet-Runner Opens Very Small 'Change_Schedulers' Window on Solaris	01-JUL-1997	Terry Koyama
SCR #1040	SmartNet Client hangs	01-JUL-1997	Terry Koyama
SCR #857	Runner001.aix.01 Step 4 New .argset doesn't show up until restarting r	15-JUL-1997	Wanda Lam
AI #64	PDC Model database	15-JUL-1997	Mike Godfrey
SCR #1141	Add option to enable/disable learning in SN-submit	15-JUL-1997	Marc Weissman
SCR #1139	Improve performance of findEarliestFit function and update of unit test	15-JUL-1997	Francesca Mirabil

SmartNet Team Report
PAGE: 3

S C R s / A I s O p e n e d

Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:46:22 PDT 1997

SCR/AI #	SCR Title	Date Opened	Status	Assignee
SCR #1103	Minor modifications for NASA planning	07-MAR-1997	CLOSED	Matt Kussow
SCR #1104	HP CC -O compiler crash on utils.cc	07-MAR-1997	CLOSED	Matt Kussow
SCR #1107	Implement new asynchronous API (Client and Server)	07-MAR-1997	OPEN	John Lima
SCR #1108	Job network profile information not getting to scheduler	07-MAR-1997	CLOSED	Matt Kussow
SCR #1100	sn-ap needs to handle jobs that are already running.	07-MAR-1997	CLOSED	Marc Weissma
SCR #1110	sn-submit006 step 2 Failed to complete Successfully on any Platform.	07-MAR-1997	CLOSED	John Lima
SCR #1111	test ap001 failed	07-MAR-1997	CLOSED	Marc Weissma
SCR #1112	ap021 failed to read from a log file (invalid format error)	17-MAR-1997	CLOSED	Marc Weissma
SCR #1116	Minor modifications for NASA planning II	17-MAR-1997	CLOSED	
SCR #1113	setup new style SN_ VERSION for main devel, and v2_7_a branch	17-MAR-1997	CLOSED	
SCR #1115	take libdebug stuff out Of libmrh	17-MAR-1997	CLOSED	
SCR #1114	correct improper for statement in sn/server/src/Model.cc	17-MAR-1997	CLOSED	

SmartNet Team Report
PAGE: 1

S C R s / A I s O p e n e d

Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:46:22 PDT 1997

SCR/AI #	SCR Title	Date Opened	Status	Assignee
SCR #1003	SmartNet-Runner Taking Longer than 2 Seconds to Load an Argument Set	15-APR-1997	CLOSED	Matt Kussow
SCR #1040	SmartNet Client hangs	15-APR-1997	CLOSED	Terry Koyama
SCR #1128	sn-submit fails to use machineLogin.db	15-APR-1997	CLOSED	Matt Kussow
SCR #1132	Constraint's scheduler combine load enhancements	15-APR-1997	CLOSED	Elaine Keith
SCR #1121	runner001 step 5 failed On all 3 platforms.	15-APR-1997	CLOSED	Wanda Lam
SCR #1119	create a network agent For SmartNet	15-APR-1997	OPEN	Marc Weissma
SCR #1124	ETC not calculated Correctly with runner	15-APR-1997	CLOSED	Matt Kussow
SCR #1117	FileBox not working Under GCC 2.7	15-APR-1997	CLOSED	John Lima
SCR #1109	Too much extraneous SNAP output.	15-APR-1997	CLOSED	Marc Weissma
SCR #1071	asynccomm010 failed on SunOS	15-APR-1997	CLOSED	Terry Koyama
SCR #1130	conversion routine Chokes on asynccomm messages.	15-APR-1997	CLOSED	Marc Weissma
SCR #1081	Editor001 Test COMMIT Button Does Not Ungray	15-APR-1997	CLOSED	Terry Koyama
SCR #1098	SmartNet Log File and Reader headers are allocated whenever included.	15-APR-1997	CLOSED	Marc Weissma

SmartNet Team Report
PAGE: 2

S C R s / A I s O p e n e d
Between 01-MAR-1997 and 05-SEP-1997

Fri Sep 5 09:46:22 PDT 1997

SCR/AI #	SCR Title	Date Opened	Status	Assignee
SCR #1133	Simplify server Override mechanism	12-JUN-1997	OPEN	Matt Kussow
SCR #1134	Improved network topology representation	12-JUN-1997	OPEN	Francesca Mi
SCR #1135	sn-ap's LIVE command line option is useless and needs to be modified.	12-JUN-1997	CLOSED	Terry Koyama
SCR #1136	Replace all cerr's, Cout's and other output statements with debug and	17-JUN-1997	CLOSED	Terry Koyama
SCR #1141	Add option to enable/ Disable learning in SN-submit	01-JUL-1997	CLOSED	Marc Weissma
SCR #1140	"sn-ap -d All -s -f New.snl -play" produces File name error.	01-JUL-1997	CLOSED	Terry Koyama
SCR #1138	Verbose is suppose to Use cout (rather than cerr) as its output stream	01-JUL-1997	CLOSED	Terry Koyama
SCR #1139	Improve performance of findEarliestFit function and update of unit tes	01-JUL-1997	CLOSED	Francesca Mi
SCR #857	Runner001.aix.01 Step 4 New .argset doesn't show up until restarting r	15-JUL-1997	CLOSED	Wanda Lam
SCR #1106	Upgrade to existing CPU Load agent	15-JUL-1997	OPEN	Marc Weissma
SCR #1143	smartnet-runner core Dumped when adding dependency.	15-JUL-1997	OPEN	Wanda Lam

SmartNet Team Report
PAGE: 3

APPENDIX E. TEAM POWER REPORT

T e a m P o w e r R e p o r t (Page 1) September 05, 1997

SCR/AI	Priority	% Done	Est.LOE	Curr.LOE	Last Week	ALOE
John Lima		Q4-1996(01-JAN-1997)			POE: 0.60	
1). AI#45	NORMAL	96	24	69	0	71
Hrs. Remaining: 9	Total LOE: 24	LOE Bal.: -99999	ALOEB: 3			
Matt Kussow		Q4-1996(01-JAN-1997)			POE: 0.40	
2). AI#21	NORMAL+	90	40	26	0	28
3). AI# 8	NORMAL	0	8	0	0	8
Hrs. Remaining: 9	Total LOE: 48	LOE Bal.: -19	ALOEB: -7			
Francesca Mirabile		Q3(10-SEP-1997)			POE: 0.50	
4). SCR#1134	NORMAL+	22	320	208	0	945
Hrs. Remaining: 35	Total LOE: 320	LOE Bal.: -95	ALOEB: -720			
John Lima		Q3(10-SEP-1997)			POE: 0.60	
5). SCR#1107	NORMAL+	55	1600	1721	0	3129
Hrs. Remaining: 35	Total LOE: 1600	LOE Bal.: -99999	ALOEB: -1387			
Marc Weissman		Q3(10-SEP-1997)			POE: 0.50	
6). SCR#1106	NORMAL	20	80	16	0	80
Hrs. Remaining: 35	Total LOE: 80	LOE Bal.: -47	ALOEB: -47			
Matt Kussow		Q3(10-SEP-1997)			POE: 0.40	
7). SCR#1133	NORMAL	0	40	0	0	40
Hrs. Remaining: 35	Total LOE: 40	LOE Bal.: -26	ALOEB: -26			
Wanda Lam		Q3(10-SEP-1997)			POE: 0.40	
8). SCR#1143	URGENT	80	8	6	0	7
Hrs. Remaining: 35	Total LOE: 8	LOE Bal.: 12	ALOEB: 13			

_____ _____

Date

_____

Government Representative

APPENDIX F. SCR METRICS FOR PROBLEM SCRs

Version 2.1-beta

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 11/01/94 to 02/15/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
11/06/94	0	0
11/13/94	0	0
11/20/94	0	0
11/27/94	0	0
12/04/94	0	0
12/11/94	3	3
12/18/94	6	10
12/25/94	9	10
01/01/95	13	17
01/08/95	19	17
01/15/95	21	31
01/22/95	28	38
01/29/95	23	0
02/05/95	23	5
02/12/95	21	4
2/15/95	18	5

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 11/01/94 to 02/15/95)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
11/06/94	0	0	0	0
11/13/94	0	0	0	0
11/20/94	0	0	0	0
11/27/94	0	0	0	0
12/04/94	0	0	0	0
12/11/94	1	2	6	0
12/18/94	0	0	10	0
12/25/94	0	1	2	3
01/01/95	0	0	3	1
01/08/95	0	1	0	0
01/15/95	1	0	4	1
01/22/95	0	0	0	0

01/29/95	0	0	7	2
02/05/95	2	4	2	2
02/12/95	0	6	6	2
2/15/95	0	2	10	4

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 11/01/94 to 02/15/95)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
11/6/94	0	0
11/13/94	0	0
11/20/94	0	0
11/27/94	0	0
12/4/94	0	0
12/11/94	9	0
12/18/94	10	7
12/25/94	6	4
1/1/95	4	0
1/8/95	1	4
1/15/95	6	2
1/22/95	0	0
1/29/95	9	8
2/5/95	10	6
2/12/95	14	9
2/15/95	16	4

SCR Total OPEN Report (week resolution, problems only)
(Report covers 11/01/94 to 02/15/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
11/6/94	0	0
11/13/94	0	0
11/20/94	0	0
11/27/94	0	0
12/4/94	0	0
12/11/94	9	3
12/18/94	12	2
12/25/94	14	2
1/1/95	18	2
1/8/95	15	3
1/15/95	19	2
1/22/95	19	2
1/29/95	20	0
2/5/95	24	2
2/12/95	30	2
2/15/95	42	3

Version 2.1

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 02/16/95 to 02/21/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
02/19/95	22	9
2/21/95	22	6

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 02/16/95 to 02/21/95)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
02/19/95	0	0	1	0
2/21/95	0	4	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 02/16/95 to 02/21/95)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
2/19/95	1	0
2/21/95	4	1

SCR Total OPEN Report (week resolution, problems only)
(Report covers 02/16/95 to 02/21/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
2/19/95	42	3
2/21/95	45	6

Version 2.2a

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 02/22/95 to 05/04/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
02/26/95	28	14
03/05/95	36	25
03/12/95	27	5
03/19/95	35	12
03/26/95	38	10
04/02/95	44	17
04/09/95	49	29
04/16/95	53	23
04/23/95	48	13
04/30/95	55	20
5/4/95	55	25

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 02/22/95 to 05/04/95)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
02/26/95	0	1	0	0
03/05/95	0	0	0	0
03/12/95	0	4	20	6
03/19/95	0	1	0	0
03/26/95	1	5	3	0
04/02/95	0	1	3	0
04/09/95	0	0	3	0
04/16/95	3	3	3	0
04/23/95	2	8	7	1
04/30/95	0	0	0	0
5/4/95	0	1	4	3

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 02/22/95 to 05/04/95)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
2/26/95	1	4
3/5/95	0	1
3/12/95	30	7
3/19/95	1	5
3/26/95	9	3
4/2/95	4	6
4/9/95	3	16
4/16/95	9	9
4/23/95	18	6
4/30/95	0	1
5/4/95	8	7

SCR Total OPEN Report (week resolution, problems only)
(Report covers 02/22/95 to 05/04/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
2/26/95	42	3
3/5/95	41	2
3/12/95	64	4
3/19/95	60	4
3/26/95	66	8
4/2/95	64	6
4/9/95	51	3

4/16/95	51	4
4/23/95	63	10
4/30/95	62	10
5/4/95	63	9

Version 2.3

SCR Average Age Report (week resolution, problems only)
(Report covers 05/05/95 to 05/25/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
05/07/95	58	30
05/14/95	57	47
05/21/95	49	9
5/25/95	50	14

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 05/05/95 to 05/25/95)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	RITICAL	URGENT	NORMAL	LOW
05/07/95	0	0	0	0
05/14/95	0	0	2	0
05/21/95	4	13	1	1
5/25/95	3	9	4	2

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 05/05/95 to 05/25/95)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
5/7/95	0	1
5/14/95	2	21
5/21/95	19	27
5/25/95	18	18

SCR Total OPEN Report (week resolution, problems only)
(Report covers 05/05/95 to 05/25/95)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
5/7/95	62	8
5/14/95	43	4
5/21/95	35	10
5/25/95	35	6

Version 2.5

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 05/26/95 to 02/05/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
05/28/95	53	17
06/04/95	60	25
06/11/95	68	35
06/18/95	78	32
06/25/95	88	39
07/02/95	88	24
07/09/95	95	28
07/16/95	93	22
07/23/95	115	39
07/30/95	120	46
08/06/95	134	42
08/13/95	54	12
08/20/95	50	13
08/27/95	132	57
09/03/95	126	44
09/10/95	104	30
09/17/95	121	44
09/24/95	85	58
10/01/95	92	50
10/08/95	61	57
10/15/95	67	106
10/22/95	66	59
10/29/95	74	66
11/05/95	52	17
11/12/95	45	9
11/19/95	86	21
11/26/95	98	30
12/03/95	97	38
12/10/95	104	64
12/17/95	106	71
12/24/95	113	78
12/31/95	120	85
01/07/96	99	48
01/14/96	105	81
01/21/96	114	88
01/28/96	121	95
02/04/96	122	70
2/5/96	123	71

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 05/26/95 to 02/05/96)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	RITICAL	URGENT	NORMAL	LOW
05/28/95	0	0	0	0
06/04/95	1	1	1	0

06/11/95	0	0	0	1
06/18/95	0	0	0	0
06/25/95	0	0	0	0
07/02/95	0	1	1	0
07/09/95	0	1	0	0
07/16/95	0	2	1	0
07/23/95	0	0	0	0
07/30/95	0	0	1	0
08/06/95	0	1	0	0
08/13/95	0	12	5	0
08/20/95	1	8	0	0
08/27/95	0	0	1	0
09/03/95	0	1	0	0
09/10/95	7	5	1	0
09/17/95	0	0	0	0
09/24/95	1	0	8	0
10/01/95	1	0	0	0
10/08/95	0	0	9	1
10/15/95	0	0	2	1
10/22/95	0	2	2	0
10/29/95	0	0	4	0
11/05/95	6	11	6	1
11/12/95	6	20	1	0
11/19/95	6	15	2	0
11/26/95	0	3	0	0
12/03/95	0	2	2	0
12/10/95	0	0	1	0
12/17/95	0	0	1	0
12/24/95	0	0	0	0
12/31/95	0	0	0	0
01/07/96	0	2	4	0

01/14/96	0	0	2	0
01/21/96	0	0	0	0
01/28/96	0	0	0	0
02/04/96	0	1	2	0
2/5/96	0	0	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 05/26/95 to 02/05/96)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
5/28/95	0	0
6/4/95	3	5
6/11/95	1	6
6/18/95	0	3
6/25/95	0	1
7/2/95	2	0
7/9/95	1	5
7/16/95	3	1
7/23/95	0	7
7/30/95	1	3
8/6/95	1	2
8/13/95	17	10
8/20/95	9	5
8/27/95	1	17
9/3/95	1	0
9/10/95	13	12
9/17/95	0	1
9/24/95	9	4
10/1/95	1	1
10/8/95	10	0
10/15/95	3	3
10/22/95	4	1
10/29/95	4	6
11/5/95	24	16
11/12/95	27	21
11/19/95	23	45
11/26/95	3	4
12/3/95	4	3
12/10/95	1	1
12/17/95	1	0
12/24/95	0	0
12/31/95	0	0
1/7/96	6	1
1/14/96	2	4
1/21/96	0	1
1/28/96	0	0
2/4/96	3	3
2/5/96	0	0

SCR Total OPEN Report (week resolution, problems only)
(Report covers 05/26/95 to 02/05/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
5/28/95	35	6
6/4/95	33	5
6/11/95	28	4
6/18/95	25	1
6/25/95	24	1
7/2/95	26	2
7/9/95	22	2
7/16/95	24	3
7/23/95	17	2
7/30/95	15	2
8/6/95	14	2
8/13/95	21	10
8/20/95	25	14
8/27/95	9	2
9/3/95	10	3
9/10/95	11	5
9/17/95	10	4
9/24/95	15	3
10/1/95	15	4
10/8/95	25	4
10/15/95	25	2
10/22/95	28	4
10/29/95	26	4
11/5/95	34	17
11/12/95	40	23
11/19/95	18	5
11/26/95	17	4
12/3/95	18	3
12/10/95	18	2
12/17/95	19	2
12/24/95	19	2
12/31/95	19	2
1/7/96	24	4
1/14/96	22	2
1/21/96	21	2
1/28/96	21	2
2/4/96	21	3
2/5/96	21	3

Version 2.6

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 02/06/96 to 04/22/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
02/11/96	123	59
02/18/96	136	84
02/25/96	123	91
03/03/96	122	130
03/10/96	118	93
03/17/96	132	74
03/24/96	116	42
03/31/96	81	4

04/07/96	107	8
04/14/96	122	16
04/21/96	146	25
4/22/96	147	26

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 02/06/96 to 04/22/96)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
02/11/96	2	0	2	0
02/18/96	0	0	0	0
02/25/96	0	0	2	0
03/03/96	0	0	4	0
03/10/96	0	1	1	0
03/17/96	0	0	0	0
03/24/96	0	1	6	0
03/31/96	0	16	0	0
04/07/96	0	7	2	0
04/14/96	0	1	1	0
04/21/96	0	0	2	0
4/22/96	0	0	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 02/06/96 to 04/22/96)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
2/11/96	4	3
2/18/96	0	1
2/25/96	2	2
3/3/96	4	4
3/10/96	2	0
3/17/96	0	5
3/24/96	7	4
3/31/96	16	6
4/7/96	9	16
4/14/96	2	4
4/21/96	3	7
4/22/96	0	0

SCR Total OPEN Report (week resolution, problems only)
(Report covers 02/06/96 to 04/22/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
2/11/96	22	4

2/18/96	21	3
2/25/96	21	3
3/3/96	21	2
3/10/96	23	3
3/17/96	18	1
3/24/96	21	2
3/31/96	31	13
4/7/96	24	7
4/14/96	22	4
4/21/96	18	2
4/22/96	18	2

Version 2.7

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 04/23/96 to 10/10/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
04/28/96	132	23
05/05/96	132	30
05/12/96	142	32
05/19/96	141	39
05/26/96	134	59
06/02/96	141	66
06/09/96	150	73
06/16/96	157	80
06/23/96	164	87
06/30/96	171	94
07/07/96	178	101
07/14/96	161	108
07/21/96	174	115
07/28/96	181	122
08/04/96	187	129
08/11/96	147	47
08/18/96	154	54
08/25/96	147	47
09/01/96	170	52
09/08/96	204	97
09/15/96	223	104
09/22/96	95	14
09/29/96	72	15
10/06/96	129	32
10/10/96	150	28

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 04/23/96 to 10/10/96)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
04/28/96	0	2	2	0
05/05/96	0	0	2	0

05/12/96	0	0	2	0
05/19/96	0	0	2	0
05/26/96	0	0	9	0
06/02/96	0	0	0	0
06/09/96	0	0	2	0
06/16/96	0	0	0	0
06/23/96	0	0	0	0
06/30/96	0	0	0	0
07/07/96	0	0	0	0
07/14/96	0	0	3	0
07/21/96	0	0	0	0
07/28/96	0	0	0	0
08/04/96	0	0	1	0
08/11/96	0	2	2	0
08/18/96	0	0	0	0
08/25/96	0	1	1	0
09/01/96	0	1	1	0
09/08/96	0	0	0	0
09/15/96	0	0	1	0
09/22/96	4	12	4	0
09/29/96	0	9	5	0
10/06/96	0	2	0	0
10/10/96	1	11	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 04/23/96 to 10/10/96)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
4/28/96	4	1
5/5/96	2	3
5/12/96	2	3
5/19/96	2	1
5/26/96	9	8
6/2/96	0	0
6/9/96	2	3
6/16/96	0	0

6/23/96	0	0
6/30/96	0	0
7/7/96	0	0
7/14/96	3	0
7/21/96	0	1
7/28/96	0	0
8/4/96	1	1
8/11/96	4	6
8/18/96	0	0
8/25/96	2	0
9/1/96	2	7
9/8/96	0	3
9/15/96	1	2
9/22/96	20	1
9/29/96	14	0
10/6/96	2	22
10/10/96	12	18

SCR Total OPEN Report (week resolution, problems only)
(Report covers 04/23/96 to 10/10/96)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
4/28/96	21	3
5/5/96	20	3
5/12/96	19	2
5/19/96	20	2
5/26/96	21	1
6/2/96	21	1
6/9/96	20	1
6/16/96	20	1
6/23/96	20	1
6/30/96	20	1
7/7/96	20	1
7/14/96	23	1
7/21/96	22	1
7/28/96	22	1
8/4/96	22	1
8/11/96	20	3
8/18/96	20	3
8/25/96	22	4
9/1/96	17	4
9/8/96	14	2
9/15/96	13	2
9/22/96	32	18
9/29/96	46	27
10/6/96	26	12
10/10/96	20	9

Version 2.7.2

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 10/11/96 to 03/05/97)

Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
10/13/96	153	31
10/20/96	257	93
10/27/96	264	100
11/03/96	271	107
11/10/96	278	114
11/17/96	217	89
11/24/96	253	37
12/01/96	260	44
12/08/96	166	51
12/15/96	173	58
12/22/96	180	65
12/29/96	187	72
01/05/97	194	79
01/12/97	302	86
01/19/97	309	93
01/26/97	316	100
02/02/97	323	107
02/09/97	330	114
02/16/97	337	121
02/23/97	344	128
03/02/97	351	135
3/5/97	354	138

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 10/11/96 to 03/05/97)
Date: 11/17/97

	SCR Priorities			
Date Week Ending	CRITICAL	URGENT	NORMAL	LOW
10/13/96	0	0	0	0
10/20/96	0	6	0	0
10/27/96	0	0	0	0
11/03/96	0	0	0	0
11/10/96	0	0	0	0
11/17/96	0	7	5	0
11/24/96	0	2	5	1
12/01/96	0	0	0	0
12/08/96	0	0	10	0
12/15/96	0	0	0	0
12/22/96	0	0	0	0
12/29/96	0	0	0	0
01/05/97	0	0	0	0
01/12/97	0	0	0	0

01/19/97	0	0	0	0
01/26/97	0	0	0	0
02/02/97	0	0	0	0
02/09/97	0	0	0	0
02/16/97	0	0	0	0
02/23/97	0	0	0	0
03/02/97	0	0	0	0
3/5/97	0	0	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 10/11/96 to 03/05/97)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
10/13/96	0	0
10/20/96	6	14
10/27/96	0	0
11/3/96	0	0
11/10/96	0	0
11/17/96	12	9
11/24/96	8	13
12/1/96	0	0
12/8/96	10	4
12/15/96	1	0
12/22/96	0	0
12/29/96	0	0
1/5/97	0	0
1/12/97	1	9
1/19/97	0	0
1/26/97	0	0
2/2/97	0	0
2/9/97	0	0
2/16/97	0	0
2/23/97	0	0
3/2/97	0	0
3/5/97	0	0

SCR Total OPEN Report (week resolution, problems only).
(Report covers 10/11/96 to 03/05/97)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
10/13/96	20	9
10/20/96	12	3
10/27/96	12	3
11/3/96	12	3
11/10/96	12	3
11/17/96	15	3
11/24/96	10	1
12/1/96	10	1
12/8/96	16	1

12/15/96	16	1
12/22/96	16	1
12/29/96	16	1
1/5/97	16	1
1/12/97	9	1
1/19/97	9	1
1/26/97	9	1
2/2/97	9	1
2/9/97	9	1
2/16/97	9	1
2/23/97	9	1
3/2/97	9	1
3/5/97	9	1

Version 3.0

SCR Metrics for Problem SCRs
==========

SCR Average Age Report (week resolution, problems only)
(Report covers 03/06/97 to 10/01/97)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
03/09/97	358	142
03/16/97	365	149
03/23/97	336	156
03/30/97	343	163
04/06/97	350	170
04/13/97	357	177
04/20/97	348	94
04/27/97	355	101
05/04/97	356	108
05/11/97	363	115
05/18/97	370	122
05/25/97	377	129
06/01/97	341	136
06/08/97	348	143
06/15/97	349	150
06/22/97	419	247
06/29/97	426	254
07/06/97	550	0
07/13/97	557	0
07/20/97	463	5
07/27/97	470	12
08/03/97	569	19
08/10/97	576	26
08/17/97	583	33
08/24/97	590	40
08/31/97	597	47
09/07/97	604	54
09/14/97	611	61
09/21/97	618	68
09/28/97	625	75
10/1/97	628	78

SCR OPEN Activity Report (week resolution, problems only)
(Report covers 03/06/97 to 10/01/97)
Date: 11/17/97

Date Week Ending	SCR Priorities CRITICAL	URGENT	NORMAL	LOW
03/09/97	0	2	0	0
03/16/97	0	0	0	0
03/23/97	0	0	1	0
03/30/97	0	0	0	0
04/06/97	0	1	0	0
04/13/97	0	0	0	0
04/20/97	1	0	10	0
04/27/97	0	0	0	0
05/04/97	0	0	1	0
05/11/97	0	0	0	0
05/18/97	0	0	0	0
05/25/97	0	0	0	0
06/01/97	0	0	1	0
06/08/97	0	0	0	0
06/15/97	0	0	1	0
06/22/97	0	0	0	1
06/29/97	0	0	0	0
07/06/97	0	0	1	1
07/13/97	0	0	0	0
07/20/97	0	1	2	0
07/27/97	0	0	0	0
08/03/97	0	0	0	0
08/10/97	0	0	0	0
08/17/97	0	0	0	0
08/24/97	0	0	0	0
08/31/97	0	0	0	0
09/07/97	0	0	0	0
09/14/97	0	0	0	0
09/21/97	0	0	0	0

09/28/97	0	0	0	0
10/1/97	0	0	0	0

SCR Open/Close Rate Report (week resolution, problems only)
(Report covers 03/06/97 to 10/01/97)
Date: 11/17/97

Date Week Ending	OPENed	CLOSED
3/9/97	5	3
3/16/97	0	0
3/23/97	4	5
3/30/97	0	0
4/6/97	1	1
4/13/97	0	0
4/20/97	14	3
4/27/97	0	0
5/4/97	1	6
5/11/97	0	0
5/18/97	0	0
5/25/97	0	0
6/1/97	1	3
6/8/97	0	0
6/15/97	1	1
6/22/97	1	4
6/29/97	0	0
7/6/97	2	9
7/13/97	0	0
7/20/97	3	2
7/27/97	0	0
8/3/97	0	1
8/10/97	0	0
8/17/97	0	0
8/24/97	0	0
8/31/97	0	0
9/7/97	0	0
9/14/97	0	0
9/21/97	0	0
9/28/97	0	0
10/1/97	0	0

SCR Total OPEN Report (week resolution, problems only)
(Report covers 03/06/97 to 10/01/97)
Date: 11/17/97

Date Week Ending	All SCRs OPENed	CRITICAL/URGENT SCRs
3/9/97	9	1
3/16/97	9	1
3/23/97	10	1
3/30/97	10	1
4/6/97	10	1
4/13/97	10	1
4/20/97	10	2
4/27/97	10	2
5/4/97	8	2
5/11/97	8	2
5/18/97	8	2
5/25/97	8	2

6/1/97	9	2
6/8/97	9	2
6/15/97	9	2
6/22/97	7	1
6/29/97	7	1
7/6/97	5	0
7/13/97	5	0
7/20/97	6	1
7/27/97	6	1
8/3/97	5	1
8/10/97	5	1
8/17/97	5	1
8/24/97	5	1
8/31/97	5	1
9/7/97	5	1
9/14/97	5	1
9/21/97	5	1
9/28/97	5	1
10/1/97	5	1

APPENDIX G. VERSION DESCRIPTION DOCUMENT

```
                VERSION DESCRIPTION DOCUMENTS
************************************************************
2.6
************************************************************
Version:    2.6
Lockdown Date:    08-MAR-96   Actual:     18-MAR-96
Release Date:     29-MAR-96   Actual:     22-APR-96

Status key:
      1 - ill defined (unclear what is needed, or scope is to broad)
      2 - ready for detailed design meetings
      3 - ready for final design
      4 - ready to be implemented (minimal design work remaining,
             may denote relatively small code changes)
      5 - completed, ready for final testing

Customers/Projects
----------------
Highest priority:
    NSA
    John Schill
      NSS
      Mahen - Planning (pre-positioning)
      EADSIM
      JTF-ATD

Lesser priority: NASA, JMCIS, IBM/Cray, Boeing, NIH, NCAR, Bob Lucas

Requirements - implementation for March release:
-----------------------------------------------
Status                                  SCRs
======                                  ====
            Highest priority
5   - Generational scheduling algorithm
      754-C

5   - Dynamic information (network/cpu/ memory, disk space, etc)  (ALL)
          includes the use of this information in schedulers???
      64    612-C

5   - Improved learning/Black hole problem (All)
      760-C 158-H

5   - Complete Reimplementation of schedule monitor
      fix performance problems/scalability, make it replay like,
      add admin features???
      761-C 766   376   635   491   72    73    314

5   - Rogue/Long running jobs (Client querying)  (NSA)
      758-C 661-C
4   - Linux port
      634

5   - Intra-User Security
      only work involving limiting job control to owning client
      752-C
```

5 - CORBA Interface (JTF)
Resulting implementation is a CORBUS wrap which is not directly part of the release
771

Lesser priority

5 - Easier DB configuration/manipulation/merging (ALL)
cleanup existing utility scripts and bring into release/manuals
764-C

3 - Command line clients (sn-submit, sn-control, etc)
mainly changes to runtime library and use of that library by sn-submit
789-C 689-C 630 79 391-H

5 - Persistence/survivability of service (NSA)
only remove existing out of date recovery mechanism
762-C

2 - API Reimplementation
only complete design
765

Special cases (optional or TBD)

3 - Improvements to sn-replay
384-R 616-R 618-R 619-R 766-R

3 - MetaRMS (NSA, NSS)
NONE

1 - Planning (pre-positioning) (JTF, NASA)
NONE

1 - SmartNet-Commserver interaction
you mentioned this a couple of times, Richard, but I still need clarification.
NONE

2 - Generic user "data" field in jobinfo struct (JTF, NSA)
someone needs to convince me why we want to do this
NONE

Requirements - Research emphasis:

Highest priority (hope to have some in Sept. release)

- Multitasking (JTF, JMCIS, IBM/Cray)
- Constraints Scheduling (Time, memory, latency, bandwidth) (JTF)
- Co-dependency between processes (JTF-ATD)
- Handling Interactive Jobs (ALL)
- Reservations (jobs, alternate usage, resource maintenance times) (JTF, NSA)
- Multi-resource scheduling (subprocesses, machine clusters, networks) (JTF)
- Auto-Partitioning (subprocesses or machine clusters)
- Multiple scheduling criteria (specifically user latency) (ALL)
- API Reimplementation

- More complete scheduling choices matrix (better modularization of schedulers???)

Lesser priority
- Server cooperation
- System security
- Hard Priorities and overrides (NSA)
- Process migration
- DCE/RMS supported integration (NSA, NASA, NSS)
- Automatic definition of non-DB defined jobs (JTF, Cray, IBM).
- Master startup client
- Better installation/setup procedures
- Stable schedules and measurements (NASA)
- Multiuser scheduling interleaving (JTF)
- Rescheduling too often.
- Global overrides are not set-able through the DB file.

These next few (last, but not least on Mark's list) will automatically be addressed somewhat in this release. I definitely want to take a peek at our current performance capabilities (size of matrix, number of clients). However, none of them will be a primary focus of effort for this release. Testing would be the exception. If anyone had a plan for something that would significantly improve our testing process, it would really help our bug and manpower problems.

- Improved Testing
- Fewer Bugs
- Performance issues
- Usability
- Improved Documentation

All other GUI modifications would be limited to fixes dictated by the other requirements (listed above), minor bug fixes, and minor usability enhancements. Most of these would be to the runner, maybe a few to the editor, and probably none to the VHM monitor.

****

2.7

****

Version: 2.7
Lockdown Date: 03-SEP-96 Actual: 10-SEP-96
Release Date: 01-OCT-96 Actual: 10-OCT-96

Status key:
1 - ill defined (unclear what is needed, or scope is to broad)
2 - ready for detailed design meetings
3 - ready for final design
4 - ready for implementation
5 - ready for initial testing
6 - completed, ready for final testing

Customers/Projects

BC2A JTF NSA PCD NASA NSS

Requirements - implementation for October release:

		Hrs	S
		===	=
1.	Finish initial SNAP implementation. Complete the new logger, and integrate with SNAP.	160	6

Dep: SCR: 53(a)

2. API Reimplementation. 160 6
Dep: SCR: 765

3. Scheduling around a fixed schedule. Includes admin control 120 6
through SNAP.
Dep: SCR: 462 463

4. Profiling and Optimization of server and schedulers. 80 6
Dep: SCR: 45(a)

5. Reimplementation of runtime library into c++ 80 6
Dep: SCR: 630 886 791

6. Implement user selectability of secondary subschedulers 80 6
(for Meta and Generational schedulers)
Dep: SCR: 926

7. Dynamic information - handle network latency information 32 6
Dep: SCR: 920

Continuing research not completed in this release
---

8. Handle multiple, job based, datahosts in
scheduling 160 3
Dep: SCR: 776 777

9. Workflow scheduling (scheduling on criteria other than 120 3
Eta score)
Dep: SCR:

10. MetaRMS (NSA, NSS) 160 3
Dep: SCR:

11. Data staging for BC2A. This involves setting up an RMS
that keeps 160 2
track of the disk space available on a machine and manages
removal of old/low-priority files in favor of new/
high-priority ones.
Dep: SCR:

Items not included in this release
---

12. Implement automatic backup server capability 160 2
Dep: SCR: 929

13. Remove FomMatrix from server. This involves storing
scheduling 80 3
information (ETCs) in a job centric manner.
Dep: SCR: 928

14. Integration with ISIS and HORUS DCEs (NSA, NASA) 120 2
Dep: SCR:

Topics of Research Interest

- Multitasking (JTF, JMCIS, IBM/Cray)
- Constraints Scheduling (Time, memory, latency, bandwidth) (JTF)
- Co-dependency between processes (JTF-ATD)
- Handling Interactive Jobs (ALL)
- Reservations (jobs, alternate usage, resource maintenance times) (JTF, NSA)
- Multi-resource scheduling (subprocesses, machine clusters, networks) (JTF)
- Auto-Partitioning (subprocesses or machine clusters)
- Multiple scheduling criteria (specifically user latency) (ALL)
- More complete scheduling choices matrix (better modularization of schedulers???)
- System security
- Hard Priorities and overrides (NSA)
- Process migration
- Automatic definition of non-DB defined jobs (JTF, Cray, IBM).
- Master startup client
- Better installation/setup procedures
- Stable schedules and measurements (NASA)
- Multiuser scheduling interleaving (JTF)
- Rescheduling too often.
- Global overrides are not set-able through the DB file.
- Develop Java api/client/server

Continuing goals/issues

- Improved Testing
- Fewer Bugs
- Performance issues
- Usability
- Improved Documentation

LIST OF REFERENCES

1. General Accounting Office Report GAO/NSIAD-93-198, *Test and Evaluation: DOD Has Been Slow in Improving Testing of Software-Intensive Systems*, September 1993.
2. Dr. John Osmundson, *Handouts and Notes from Naval Postgraduate School, Software Engineering and Management,* IS4300, September 1996.
3. Carnegie Mellon University, Software Engineering Institute, *The Capability Maturity Model: Guidelines for Improving the Software Process*, 1995.
4. SSC-SD SEPO Office, Software Process Improvement Overview, 22 August 1996, SEPO SPI Homepage @ http://sepo.nosc.mil/spipage.
5. Department of the Air Force, Software Technology Support Center, *Guidelines for Successful Acquisition and Management of Software-Intensive Systems*, June 1996, Volume 1.
6. Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, *Capability Maturity Model for Software, Version 1.1,* February 1993.
7. Rubin, Howard, Editor, *IT Metrics Strategies*, May 1996, Volume II, No. 5.
8. Dr. James Emery, *The Case for an Adaptive Software Development Process*, October 1997.
9. Dr. James Emery, *Adopting a New Software Development Paradigm: A Strategic Imperative.*
10. George Yamamura, Gary B. Wigle, Boeing Defense & Space Group, *SEI CMM Level 5: For the Right Reasons*, Crosstalk, The Journal of Defense Software Engineering, August 1997.
11. Kimsey M. Fowler Jr. , Boeing Defense & Space Group, *SEI CMM Level 5: A Practitioner's Perspective,* Crosstalk, The Journal of Defense Software Engineering, September 1997.
12. George Yamamura, Boeing Defense & Space Group, *World Class Practices of Boeing's Space Transportation Systems Organization,* The Software Technology Conference, April 1997.
13. Harvard Business School, *Microsoft Corporation: Office Business Unit*, 9-691-033, May 31, 1994.

14. SSC-SD SEPO Office, *Draft of SSC-SD Instruction for Software Engineering Process Policy,* 1 November 1996.

15. *SmartNet Brochure*, TD 2882.

16. Thomas McGibbon, Kaman Sciences Corp., *A Business Case for Software Process Improvement, A Data & Analysis Center for Software (DACS) State-of-the-Art Report*, 30 September 1996.

17. Judith G. Brodman and Donna L. Johnson, *Return on Investment (ROI) from Software Process Improvement at Measured by US Industry, 1995.*

BIBLIOGRAPHY

Boehm, Barry W., *Software Engineering Economics*, 1981.

Curtis, Bill, *Building a Cost-Benefit Case for Software Process Improvement*, TeraQuest Metrics, Inc, 1995.

Emam, Khaled El, *Software Process Newsletter*, IEEE Computer Society TCSE, No. 7, Fall 1996.

Emery, James C., *Management Information Systems, The Critical Strategic Resource,* Oxford University Press, 1987.

Emery, James C., *Cost/Benefit Analysis of Information Systems*, The Society for Management Information Systems Workshop Report No. 1.

Gilb, Tom, *The Planguage Method, A Handbook for Advanced Practical Management and Engineering Methods for Leading-Edge Competitiveness and A Guide to Critical Thinking about Complex Ideas, Problems or Systems Project Language, Process Language, Planning Language,* Version 0.3, September 14, 1996.

Herbsleb, James, Carleton, Anita, Rozum, James, Siegel, Jane, Zubrow, David, *Benefits of CMM-Based Software Process Improvement: Initial Results*, Software Engineering Institute, August 1994

Humphrey, Watts S., *Managing the Software Process*, August 1990.

Pressman, Roger S., *Software Engineering, A Practitioner's Approach*.

Science Applications International Corporation (SAIC), *Common Approach to Software Development and Maintenance*, Version 3, September 11, 1996.

SSC-SD SEPO Office, *Software Process Improvement Representatives Introductory Training (SPIRIT) tool box materials provided by the SSC-SD Software Engineering Process Office (SEPO)*, 1997.

INITIAL DISTRIBUTION LIST

No. of copies

1. Defense Technical Information Center.....2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Dr. James Emery.....1
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

4. Dr. Dan Boger, Acting Chairman, Code CS.....1
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

5. Elizabeth Gramoy, D13.....2
   Space and Naval Warfare Systems Center, San Diego
   53560 Hull Street
   San Diego, CA 92152-5001

6. Richard Freund, D42.....1
   Space and Naval Warfare Systems Center, San Diego
   53560 Hull Street
   San Diego, CA 92152-5001

7. Karen D. Prenger, D412.....2
   Space and Naval Warfare Systems Center, San Diego
   53560 Hull Street
   San Diego, CA 92152-5001

6 483NPG TH 2760
10/99 22527-200 NULE

DUDLEY KNOX LIBRARY
3 2768 00366724 7